TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Thomas Johann Seebecki elektroonikainstituut

IEE70LT

Kaarel Soon 111803

3D LASERSKÄNNERITE UURIMINE

Magistri lõputöö

Olev Märtens

Tehnikateaduste doktor

Juhtivteadur

Tallinn 2014

Autorideklaratsioon

Olen koostanud antud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud. Käesolevat tööd ei ole varem esitatud kaitsmisele kusagil mujal.

Autor: Kaarel Soon

Kuupäev:

Annotatsioon

Töö eesmärgiks on uurida kahe laserskänneri võimalusi, millega oleks võimalik kujutada ruumi 3-mõõtmelisena. Testida mõlema seadme sobivust kahe erineva rakenduse jaoks. Rakendusteks on patsiendi positsioneerimine ja teeolude kaardistamine.

Kindlaks on vaja teha, milline seade sobib kõige paremini ühele või teisele rakendusele. Esimeseks seadmeks on Kinect, mille täpsust, lahutusvõimet ja müratasset testiti objektide kujutamisel. Järgmiseks kasutati Kinect Fusionit objekti 3-mõõtmelisel kujutamisel ja suudeti eraldada koordinaadid, et kujutada objekti ristlõige kindlast kohast. Teiseks seadmeks on laserjoone kiirgaja XIMEA kaameraga. Eeltöödeldi kaamera kaadreid Gaussi silumisega müra vähendamiseks, leiti laserjoon HSV ja RGB osadega kaadrist ning eraldati laserjoone kuju Hough teisendusega, mis leiab jooned pildil. Lõpetuseks laserjoone pikslite koordinaadid teisendati ja leiti Z-koordinaat, et kujutada objekti ristlõige graafikul.

Testidest ja pilditöötlusest järeldati, et Kinect on kõige sobivam seade patsiendi positsioneerimise rakenduse jaoks. Kinecti sügavuse lahutusvõime on 1[mm] või vähem, millega on võimalik saavutada rahuldavaid tulemusi objekti 3-mõõtmelises kujutamises. Abiks on Kinect Fusion, mille 3-mõõtmelistest mudelitest on võimalik eraldada koordinaadid, mis on vajalikud rakendusele. Kinectiga kaasa tulev tarkvara on kasulikuks hüppelauaks oma tarkvara koostamisel rakendustele, mis vajavad 3-mõõtmelist kujutamist.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 56 leheküljel, 6 peatükki, 43 joonist, 2 tabelit.

Abstract

RESEARCHING 3D LASER SCANNERS

The purpose of this thesis is to study the possibilities of two different laserscanners. They would enable to project space in three dimensions. Tests need to be run in order to determine the devices' suitability for two different applications. The applications are patient positioning and mapping the conditions of a road.

Both of these applications operate in two different environments. Patient positioning is performed in a controlled environment, where noise is minimum. Mapping conditions of a road is performed in a fast-moving environment and weather conditions such as sunlight or rain will corrupt the measured data.

First of the devices is Microsoft's Kinect. Kinect's accuracy, depth resolution and noise in depth data was tested to determine Kinect's suitability for the applications. Kinect's depth resolution was concluded to be 1[mm] or less, which was the primary parameter observed. Kinect Fusion was used to draw a 3-dimensional model of an object and coordinate data was extracted to draw a graph of the object's cross-section.

Second device is a laserline emitter combined with an XIMEA camera. The images of the camera were pre-processed with Gaussian blur to reduce noise, the laserline was extracted using HSV or RGB of an image, features were extracted using Hough transform to detect the laserline and achieve pixel coordinates. In the end pixel coordinates were converted to 3-dimensional and a graph was formed to show the object's cross-section.

Through the tests and frame processing it was concluded that Kinect is the most suitable for patient positioning after it was determined that its depth resolution is 1[mm] or less, and it is possible to extract coordinate data from Kinect Fusion's 3-dimensional model of the object. Overall Kinect provides excellent software support for 3-dimensional modeling and it is possible to create your own.

This paper is written in Estonian and includes text on 56 pages, 6 chapters, 43 schematics and 2 tables.

Lühendite ja mõistete sõnastik

FPS	*Frames-per-Second*, kaadrid sekundi kohta
VGA	*Video Graphics Array*
CMOS	*Complimentary metal-oxide-semiconductor*
USB	*Universal Serial Bus*
dB	*Decibel*, detsibell
SDK	*Source Development Kit*, lähtekoodi muutja
IDE	*Integrated Development Environment*, tarkvara keskkond
API	*Application Program Interface*
2D	*2-Dimensional*, 2-mõõtmeline
3D	*3-Dimensional*, 3-mõõtmeline
RGB	*Red-Green-Blue*
HSV	*Hue-Saturation-Value*
CAD	*Computer-Aided Design*

Sisukord

Jooniste nimekiri

Tabelite nimekiri

1. Sissejuhatus

Antud uurimus käsitleb laserseadmete võimalusi, kuidas neid kasutada 3-mõõtmelise ruumi kujutamisel, et leida erinevused objektidel koordinaatide abil. Seadmeid testitakse eesmärgiga neid rakendada kahe funktsiooni jaoks.

Viimasel ajal on väga suur roll olnud 3-mõõtmelisel kaardistamisel erinevate seadmetega. Kõiki detaile on võimalik skänneerida ning tänu 3-mõõtmelistele mudelitele neid toota väga erinevatel aladel: meditsiin, mood, ehitus, infrastruktuur. Näiteks kui pole varuosadele CAD faili, siis saaks skänneriga taastada arvuti faili, mida saab kasutada,et uuesti toota või võrrelda toodetud detaili joonisega.

Laserseadmeid uuritakse kahe rakenduse jaoks, patsiendi positsioneerimine ja teeolude kaardistamine. Patsiendi positsioneerimisel on vaja teada, mis asendis patsient samal ajal oli, kui ravi sai. Järgmise ravi ajal korrata protseduuri, et leida asendi erinevused. Teeolude kaardistamine nõuab liikuvat sõidukit, millele seadmed on kinnitatud ja seadmed peavad suurema kiiruse korral suutma kaardistada teed 3-mõõtmelises ruumis, nähes ära kõik erinevused ja vead tees. Ohuks on viimase rakenduse puhul ilmastikunähtused, muutuv keskkond ja suur kiirus.

Uurimuses käsitletakse kahte laserseadet: Kinect ning laserjoone kiirgaja koos kaameraga. Kinectiga viiakse läbi testid täpsuse, lahutusvõime, mürataluvuse hindamiseks, et ta oleks sobiv rakenduste jaoks, mis vajavad 3-mõõtmelist ruumi kujutamist. Tekib võimalus kujutada objekt 3-mõõtmelises ruumis ja tuvastada näitajad. Teiseks uurimisobjektiks on seade, mis koosneb laserjoone kiirgajast ja kaamerast. Seadme põhieesmärkideks on leida laserjoon kaamera poolt salvestatud kaadrite pealt. Suur osa on pilditöötlusel, kus siis vähendatakse müra kaadritel, eraldatakse laserjoon kaadrilt ning lõpuks märgitakse laserjoon oma pikslite koordinaatidega. Pikslite koordinaatidega on võimalik leida kolmas koordinaat Z, mis aitab lõpuks objekti ristlõikena kujutada.

Kõik katsed on uurimiseks, kas laserseadmetega on üldse võimalik märgistada ja salvestatud kaadreid töödelda, nii et saaks vajalikud andmed kahe rakenduse jaoks. Töö eesmärk on kindlaks teha, kumb seade või mõlemad on optimaalsed rakendusele.

2. Uurimuses kasutatav riistvara

Uurimustöös kasutatakse testide läbiviimiseks ja testide tõestamiseks mitmesugust riistvara, mis on abiks optimaalse skänneri leidmisel. Tähtis on teada seadmete spetsiifilisi nõudeid, nende võimekust. Infoga on võimalik kaamerad kalibreerida, et saada kõige paremad testi tulemused.

2.1 Riistvara kirjeldus

Riistvara uurimistöös hõlmab kahe erineva tööpõhimõttega laserallikat. Esimene seade kiirgab infrapunalaseri võrgustiku, millega mõõdetakse sügavusandmeid. Teine seade on laser, mis kiirgab pideva laserjoone, andes talle võime märgistada igat muutust keskkonnas, kuhu laser suunatud. Laserite töö tuleb märgistada kaamerate poolt, mida hiljem töödeldakse.

Eelnevalt mainitud esimene seade kiirgab infrapunalaseri võrgustiku, mis asetatakse ruumi teatud suuruses. Seade sisaldab ka infrapuna sensorit, mille tööülesanne on lugeda võrgustiku andmed, näiteks tuua välja erinevad objektid efektiivses alas. Andmed tuleb töödelda, teha inimesele kergesti loetavaks ja siis ka salvestada. Mis koosneb ruumi sügavusandmetest – täpsus, täpsusaste, efektiivne kaugus, müra ja lahutusvõime. Tehnoloogia on hetkel olemas ja vastab kõigile kriteeriumitele, selleks on Microsofti seade Kinect[1].

Järgmiseks on laser, mis kiirgab pideva laserjoone ning kui vedada piki pinda on võimalik näha muutusi laserjoones ehk siis pinna eripärasused märkida. Samal ajal kui laserjoont veetakse, tuleb terve protsess kaamera poolt filmida. Kasulik oleks kasutada kaamerat, mille FPS oleks vähemalt üle 30, sest see mõjutaks üleüldist kaadrite kvaliteeti ja sujuvust. Kaamera suunatakse täpselt laserjoone keskpunkti, lihtsustades andmete lugemist. Laser suunab 90° laia joone. Kaamera selle jaoks on olemasolev XIMEA kaamera[2].

2.2 Kinect

Microsofti seade töötati välja ideega täpselt jälgida ja lugeda inimese liigutusi, abistada näotuvastusel, sügavusandmete lugemisel ruumis ning salvestada ümbritsev keskkond. Kinect koosneb VGA värvikaamerast, sügavussensorist ja mitmest ühendatud mikrofonist.

Videokaamerat kasutatakse Kinectis näotuvastuse protsessis ja inimese liigestetuvastuseks. Liigeste all mõeldakse kohti, mis jagavad näiteks käe kaheks liikuvaks osaks.

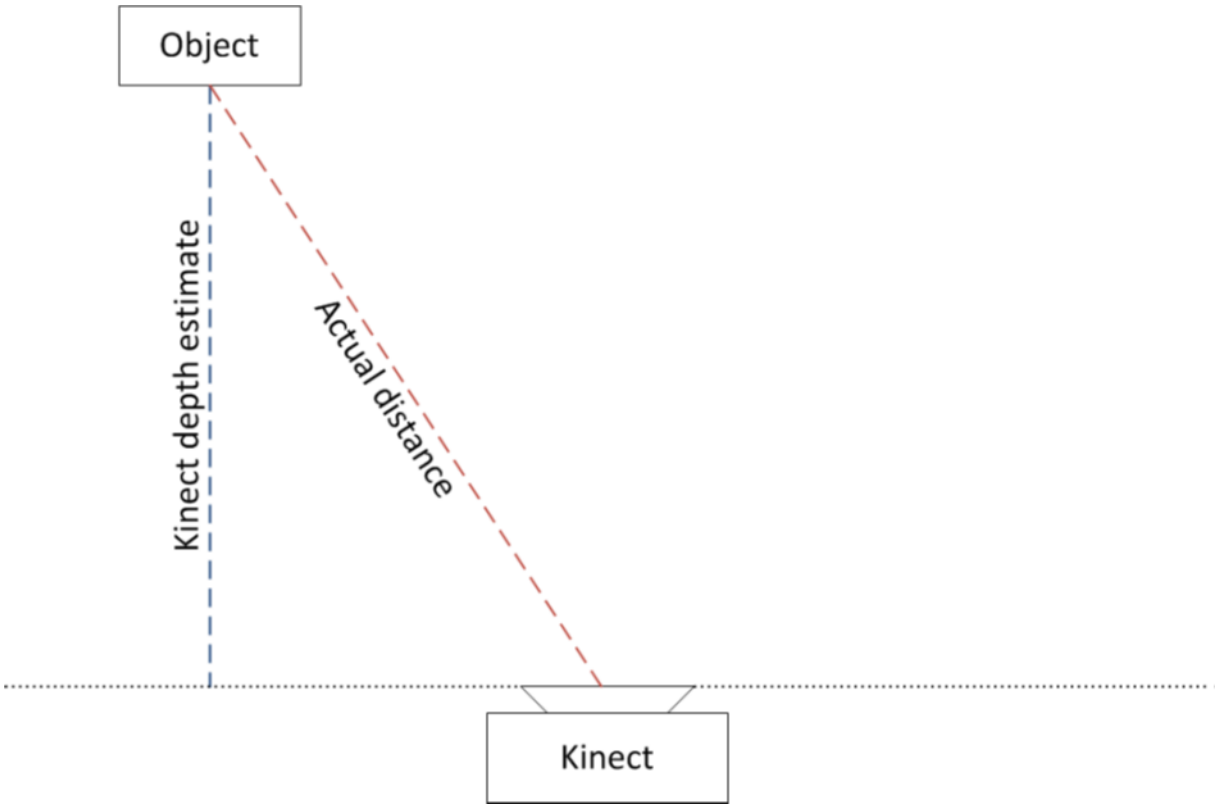
Sügavussensor koosneb infrapuna laserist ja monokromaatilisest CMOS sensorist, ning nende kahe abil on võimalik kujutada ruumi 3-mõõtmelisena.

Kinecti abil saab jälgida ja salvestada inimese liigutusi, mida nad kutsuvad skeletonjälgimiseks(skeletal tracking)[3]. Skeleton leitakse kasutaja liikuvate osade järgi, kaamera omakorda jälgib neid ruumis ja leiab nende liikumised ajas. Võimalik on tuvastada, millal kasutaja istub või seisab, kuid selle parimaks kasutamiseks peaks kasutaja vaatama Kinecti poole.

Sügavusandmed[4] koosnevad pikslitest, mille ülesanne on arvutada kõikide objektide kaugus kaamerapinnast millimeetrites. Kinectil on võimalik kasutada kolme resolutsiooni: vaikimisi väärtus 640x480, 320x240, 80x60 pikslit. Seda saab muuta tarkvara koodis. Võimalus on ka muuta minimaalset ja maksimaalset kaugust(näiteks 1 meeter – 2 meeter), seda kasutatakse, et vältida üleliigset müra ja andmeid.

Sügavusandmete tõlgendamise tööpõhimõte on näidatud (Joonis 1). Arvutatakse hinnanguline kaugus objektist kaamerapinnani ja mitte tegelik kaugus sensorist[5].

Tabel 1 annab ülevaate Kinecti tehnilistest näitajatest[6].



Joonis 1Kinecti kaugusarvutus

Tabel 1 Kinecti tehnilised näitajad

Näitaja	**Väärtus**
Kaugus	800[mm] – 4000[mm]
Horisontaalne nähtavusala	57°
Vertikaalne nähtavusala	43°
Vaikimisi resolutsioon	8-bit VGA 640 x 480
Kaadrisagedus	30Hz (30fps)
Nominaal sügavusresolutsioon (2 meetri juures)	1[cm]
Nominaal ruumilisusresolutsioon (2 meetri juures)	3[mm]
Seadme ühendus	USB + external power

2.3 XIMEA kaamera

Kaamera uurimuse jaoks on XIMEA kaamera MQ042MG-CM[7]. XIMEA kaamera on töötatud välja tööstuslike olude jaoks. Tema tehnilised näitajad on palju paremad kui tavakaameral ja on võimeline palju rohkemaks. Võimekus on väga tähtis pilditöötluses ja andmete salvestamises selles projektis. Peamine probleem on leida laserjoon ning, mida parem kaamera, seda lihtsam on andmeid töödelda. Tehnilised näitajad on tabelis 2.

Tabel 2 XIMEA kaamera tehnilised näitajad

Näitaja	**Väärtus**
Resolutsioon	2048 x 2048
Sensori tüüp	CMOS RGB Bayer Matrix
Sensori mudel	CMOSIS CMV4000
Sensori suurus	1“
Sensori aktiivala	11.27 x 11.27 [mm]
Piksli suurus	5.5 x 5.5 [µm]
Bitide arv piksli kohta	8, 10, 12
Dünaamiline ala	60dB
Kaadrisagedus	90 fps
Seadme ühendus	USB 3.0

2.4 Joonlaserid Z-Laser ZM-18 seeria

XIMEA kaamera katseteks on olemas joonlaserid Z-laser ZM-18 seeriast[32], mille kiiratud joont mõõdetakse ja töödeldakse kaadrite pealt. Laseritel on võimalik kasutada Gaussi laserit ja lineaarlaserit. Gaussi laser kasutab Gaussi jaotust ehk on intensiivsem ja kergem kaadritel leida, viga seisneb, et joon on intensiivsem keskpunktis ja hajub äärtel. Lineaarlaser ei ole nii intensiivne, kuid tal on pidev 90° raadiusega horisontaaljoon. Võimalus on laser muuta ka roheliseks, punase asemel.

3. Uurimuses kasutatav tarkvara

Kinecti tarkvara muutmiseks on Kinect SDK[8]. SDK loodi Microsofti poolt, et lihtsustada inimeste tööd, kes tahavad Kinectiga luua uusi võimalusi, kasutades kõike mida Kinect suudab pakkuda.

Pilditöötluse jaoks, mida on vaja laserjoone tuvastamiseks on olemas võimas avatud teek openCV[9]. OpenCV sisaldab funktsioone pilditöötluse ja algoritmide jaoks.

Keskkonnaks, mis suudab toetada nii SDK-d kui ka openCV-d on Microsofti enda tarkvara Visual Studio 2012[10]. See on IDE ehk integreeritud arenduskeskkond.

Lõpuks on R-keel[11], mis on programmeerimiskeel ja tarkvara keskkond. Mõeldud statistika, arvutuse ja graafika suunaga projektide jaoks. Andmete analüüsis on vajalik kasutada R-keelt.

3.1 Kinect SDK

SDK annab ligipääsu tööriistadele ja API-le, mille põhifunktsioon on arendada tarkvara Kinectile. See sarnaneb Windowsile tarkvara arendamises, kuid ta annab võimaluse kasutada elemente, mis aitavad rakendada pilditöötlust, sügavusandmeid ja skeletoni andmeid.

Mis sisaldab kolme erineva keele keskkondi, milleks on: C++, C# ja Visual Basic. Tuues kaasa ka lähtekoodi kõikides keeltes, dokumentatsiooni ja kasutajaliidese.

Kasuks on ka näited erinevate rakenduste jaoks, tehes SDK väga kasutajasõbralikuks.

3.2 OpenCV

Avatud massiivne teek openCV on välja töötatud arvutusliku efektiivsuse ja reaalajas töötavate rakenduste jaoks. Toetab keeli C, C++, Python ja Java ning töötab Windowsi, Linuxi ja MacOS keskkonnas. Tugevalt optimiseeritud ning avatud kood teevad temast ühe parema teeki, mis hetkel vabalt kättesaadav on.

Kõige suurem kasutusala sellel on arenenud pildiuurimine ja see toetab kindlaid nõudeid, mida on selles uurimustöös vaja:

- 2D ja 3D rakenduses
- Objekti tuvastus
- Liikumisest arusaamine
- Objekti leidmine liikumise ajal

Suurem osa tööst hõlmab liikuvat keskkonda ja objekti tuvastust.

3.3 Visual Studio 2012

Microsofti enda IDE, mille peamine kasutusala on tarkvara tegemine Windowsis asuvatele rakendustele, kuid on ka tugi veebilehekülgedele ja –teenustele.

Visual Studiol on mitme erineva programmeerimiskeele tugi ning kerge on nende vahel vahetada. Põhiliselt kasutatavad keeled uurimuses on C++ ja C# , openCV ja SDK jaoks vastavalt. Kuna openCV koosneb teekidest, siis Visual Studio intelligentne teekide haldaja abistab projekti läbiviimist.

4. Seadmete sobivuse uurimine

Uurimuse peamine probleem on leida parim skänner kahele rakendusele: patsiendi asukoha määramine ja teeolude kaardistamine. Tulemuseks võib ka olla mõlema skänneri sobimine rakendusele. Selle jaoks tuleb testida Kinecti tarkvara ja mõõtmistäpsust – kui täpne ja suure müraga on sügavusandmed ning jõuda järeldusele, mis rakendusele Kinect sobib.

Tuleb testida laserjoone ja XIMEA kaamera koostöötavust ning töötada välja algoritm, mille järgi teada, kuidas pilti töödelda. Idee on leida laserjoon kaadril ning see eraldada sealt ja viia 3-mõõtmelisesse ruumi.

4.1 Kinecti sobivustestid

Peatükk koosneb testidest Kinecti peal. Patsiendi asukoha määramiseks oleks vaja lahutusvõimet vähemalt 1[mm], muidu pole täpsus piisav. Teisel rakendusel – teeolude kaardistamine, kus auto kiirus võib saada määravaks ja Kinecti kaamera võib jääda nõrgaks.

Testideks on:

- Sügavusandmete põhjal lahutusvõime leidmine
- Leida täpsus ja täpsusaste
- Müra sügavusandmete hindamisel

Testid viiakse läbi erinevatel kaugustel. Testide abil leitakse vea suurus erinevatel kaugustel ja kui palju tuleks kaugust muuta, kui tahetakse leida optimaalset kaugust.

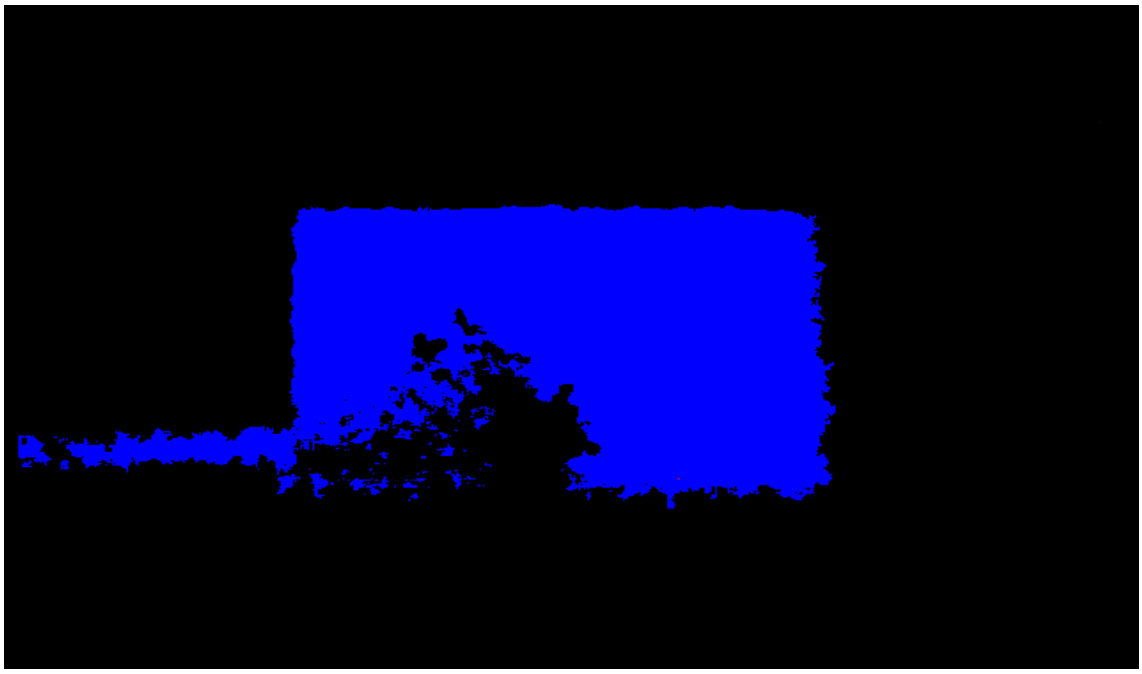
4.1.1 Kinecti lahutusvõime

Test viiakse läbi, asetades objekt kindlale kaugusele ja lastes tal logida 100 mõõtmist ühes kauguses kuni edasi liigutakse. Miinimum- ja maksimumväärtused sorteeritakse ja tuuakse välja. Graafikul esitatakse tegelike ja mõõdetud kauguste vahed.

Test esitatakse kaugustel 600[mm] – 1000[mm]. Kinecti tarkvara on modifitseeritud registreerima objekte ainult kaugusel 580[mm] kuni 1015[mm]. Objektiks on ristkülik pildi keskel.

Esimese testi kriteeriumid:

- Alustuskaugus – 600[mm]
- Lõppkaugus – 1000[mm]
- Aste – 100[mm]
- Kaadrite arv – 100
- Teek – Kinect SDK



Joonis 2 Kinecti pilt 600[mm] kaugusel

Tulemus 600 [mm] peal (Joonis 2). Näha on, et 600[mm] pole võimalik saada väga usaldusväärset pilti, sest objekt peaks olema ristkülik. Mingi osa pildist pole üldse realiseeritudki ja Kinect ei registreeri, et ta olemas oleks. Kõik mis on must asub väljaspool piire.

Joonis 3 Kinect pilt 700[mm] kaugusel

(Joonis 3) on pilt 700[mm] juures ning täpsem kujutus. Näha on pisikesi kõrvalekaldeid ja müra pildi juures. Üksik koht pole täielikult mõõdetud, kuid see võib ka olla hetkelise kaadri viga ning kuna kaadreid on 30 sekundis, siis ei mõjutaks see mõõtetulemusi märgatavalt.

Joonis 4 Kinecti pilt 800[mm] kaugusel

(Joonis 4) 800[mm] juures on ülemised ääred juba selgemad ja alumine äär on asetatud põrandale, mis võibki tekitada sellist müra, mis segab mõõtetulemusi.

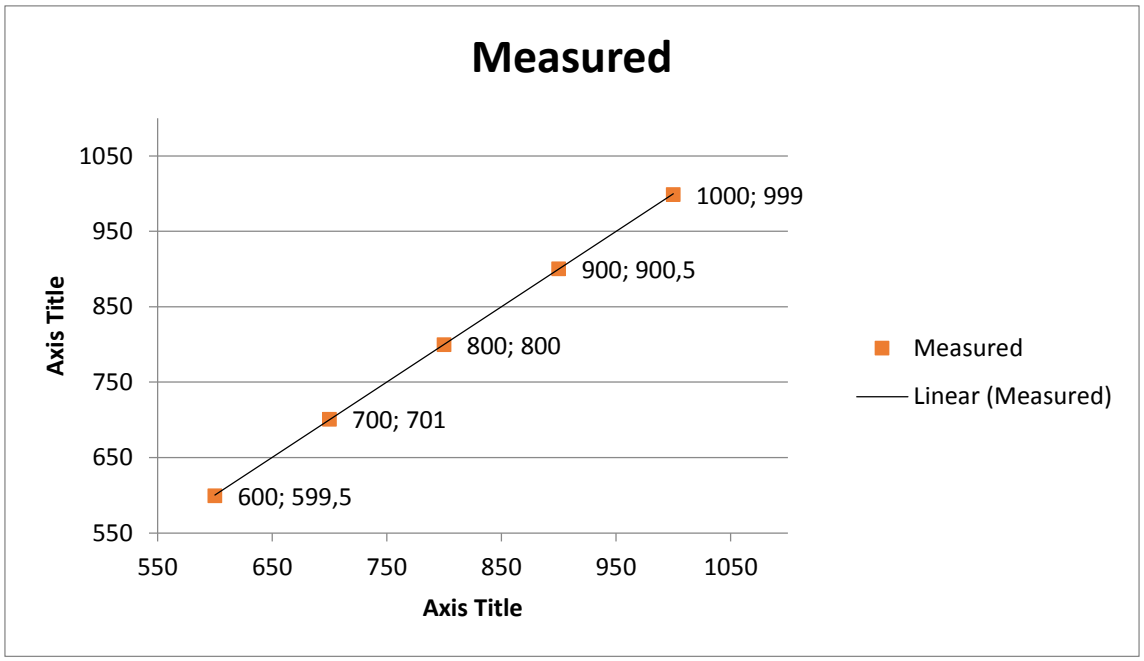
(Joonis 5) 900[mm] kaugusel on põranda müra vähemaks jäänud ja pildil pole enam suuremaid mõõtevigu silmaga näha. Tulemused esitatakse graafikul ikkagi ja saab selgema pildi.

(Joonis 6) 1000[mm] on näha kindlat ristküliku joont, kuid kaugusega suureneb äärealade müra.

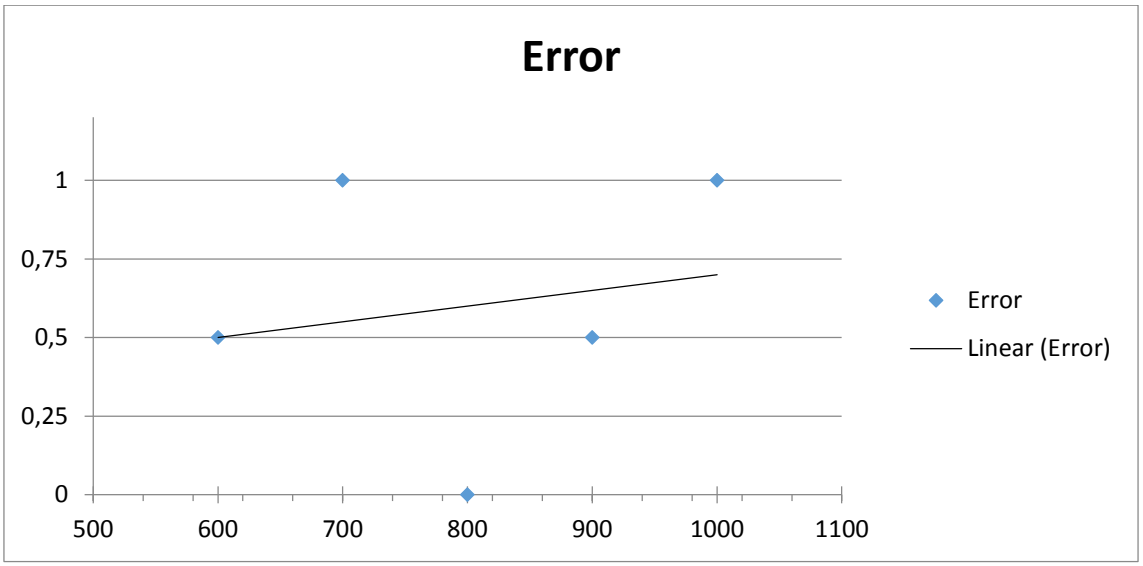
Joonis 5 Kinecti pilt 900[mm] kaugusel

Joonis 6 Kinecti pilt 1000[mm] kaugusel

Graafikul (Joonis 7) on ära toodud iga kauguse maksimaalsete ja minimaalsete väärtuste keskmised. Esimene väärtus on tegelik kaugus ja teine väärtus on keskmine.



Joonis 7 Kauguste keskmised



Joonis 8 Kauguste vead

(Joonis 8) Annab meile kauguste vead, tegelikust kaugusest ja viga ei ületa 1[mm] ning vea suurus on umber 0,7[mm].

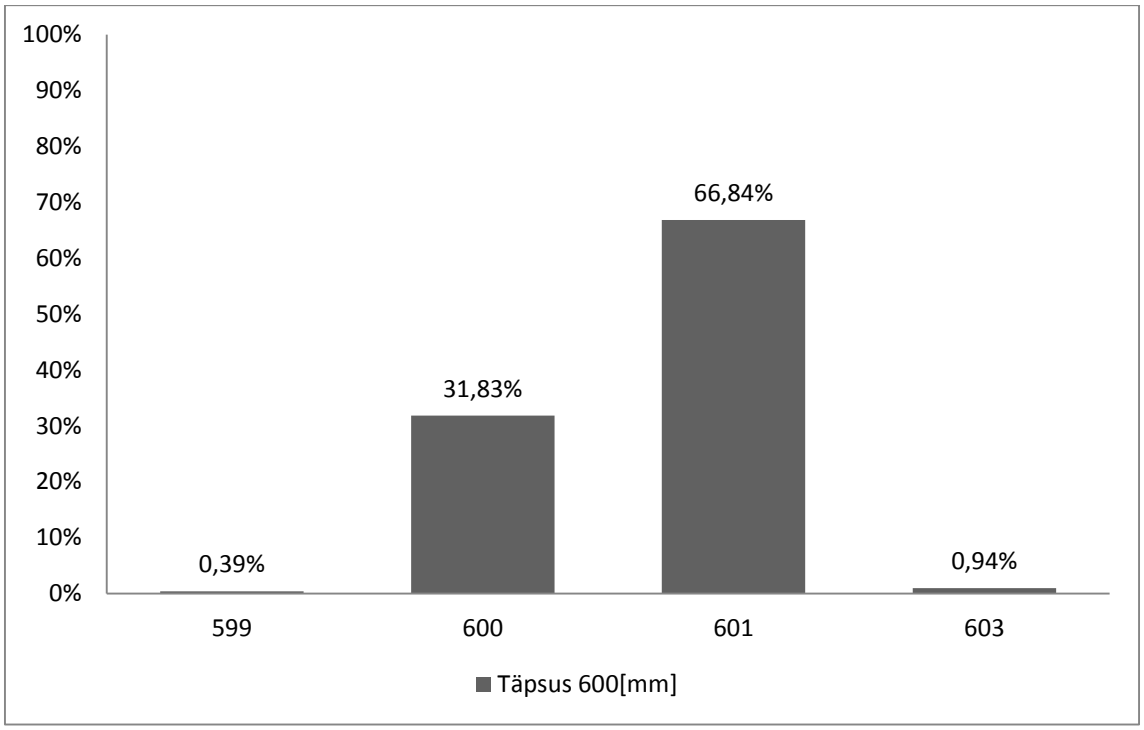
4.1.2 Kinecti täpsus

Isegi kui seade on statsionaarne on märgata piltidelt(Joonis 2-6), et pikslite väärtus on ajas muutuv. Selles testis teeb Kinect 1000 mõõtmist, kindlatel kaugustel, ühe sügavuspiksli peale. Esitatakse statistiline graafik kõige rohkem esinenud mõõtetulemustega. Graafik demonstreerib, kui palju on õiged mõõtetulemusi ning kui palju need erinevad.

Kaks testi esitatakse kaugustel 600[mm] ja 1000[mm]. Objektiks on ristkülik pildi keskel ning eelnevaid jooniseid (Joonis 2 ja Joonis 6) kasutatakse tulemuse esitamiseks.

Esimene test:

- Kaugus – 600[mm]
- Mõõtmiste arv – 1000
- Teek – Kinect SDK

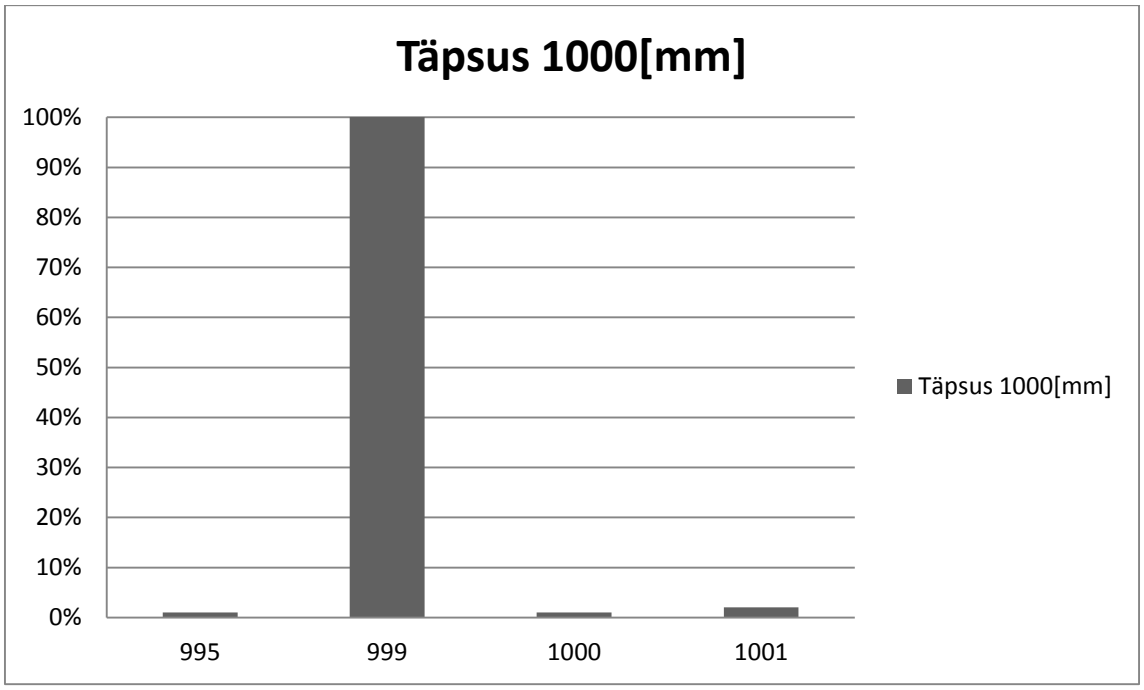


Joonis 9 Kaugusel 600[mm] täpsuse histogramm

Histogrammil on näha (Joonis 9) kindla piksli väärtust 600[mm] kaugusel. Piksli väärtus tundub varieeruvat nelja väärtuse vahel 599[mm] kuni 603[mm]. Kuid suurem osa langeb 600[mm] ja 601[mm] vahetusse lähedusse, sest teiste väärtuste osakaal on marginaalne. Kahe suurema osakaaluga väärtuste vahe on alla 1[mm].

Teine test:

- Kaugus – 1000[mm]
- Mõõtmiste arv – 1000
- Teek – Kinect SDK



Joonis 10 Kaugusel 1000[mm] täpsuse histogramm

Histogrammil on näha (Joonis 10) kindla piksli väärtust 1000[mm] kaugusel. Piksli väärtus tundub varieeruvat nelja väärtuse vahel 995[mm] kuni 1001[mm]. Kuid suurim osa langeb 999[mm]. Teiste väärtuste osakaal on marginaalne. Tuleb meelde jätta, et ka võib olla inimlik viga mõõtmistes, sest tõenäosus on, et ristküliku pind pole 1000[mm] kaugusel. Võttes testi tulemused arvesse, siis lahutusvõime jääb alla või täpselt 1[mm] juurde.

4.1.3 Kinecti müra

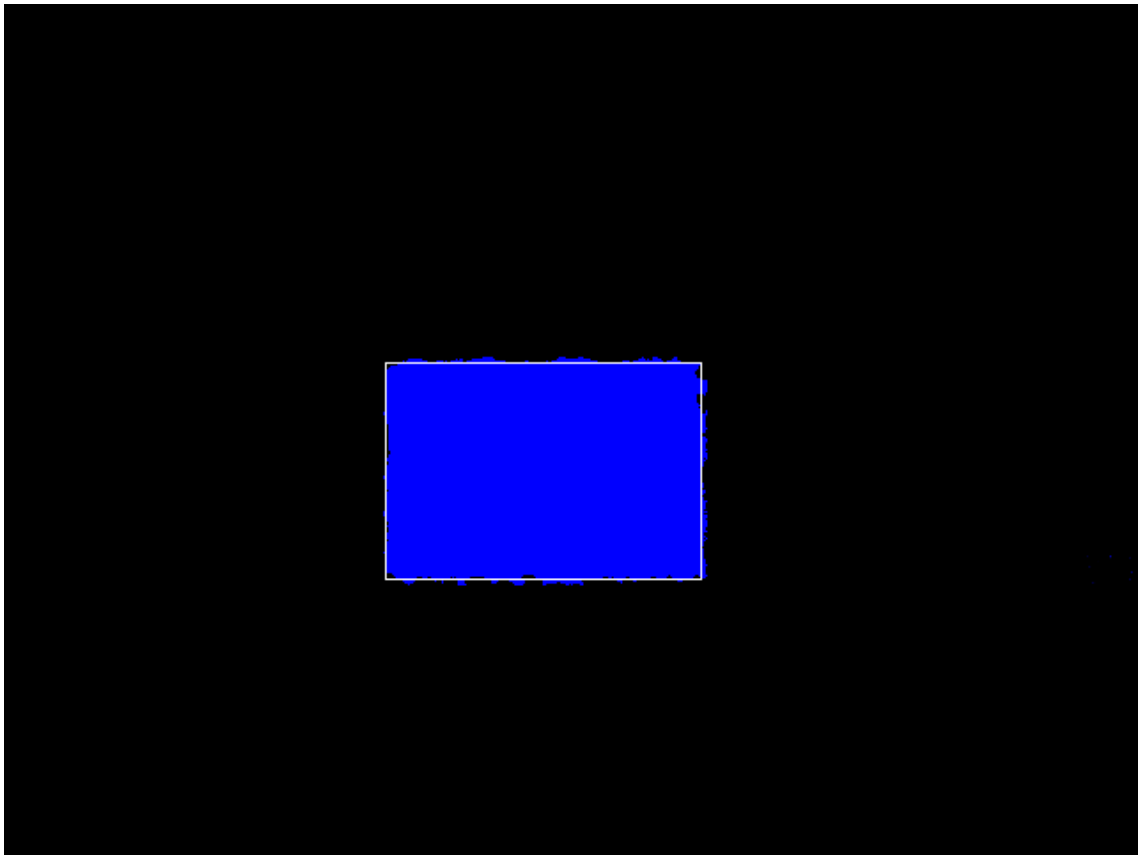
Sügavusandmete arvutamisel saame arvutatud kauguse, mis on punktist sensoralani (kirjeldatud Joonis 1), mitte punktist täpselt sensorini. Eeldades seda peaksid tulemused olema võrdsed, kui suunata Kinect siledale pinnale otse ees. Kuid arvestada tuleb ikkagi väikeste muutuste ja müraga.

Muutuste leidmiseks, salvestame 1000 kaadrit üksteise otsa et näha, kus tulemus erines. Erinevused näitavad pikslite asukoha muutusi äärtes ehk kui palju oli ääres müra ja paljud pikslid jäid tabamata.

Testis asetatakse objekt 1000[mm] kaugusele Kinectist.

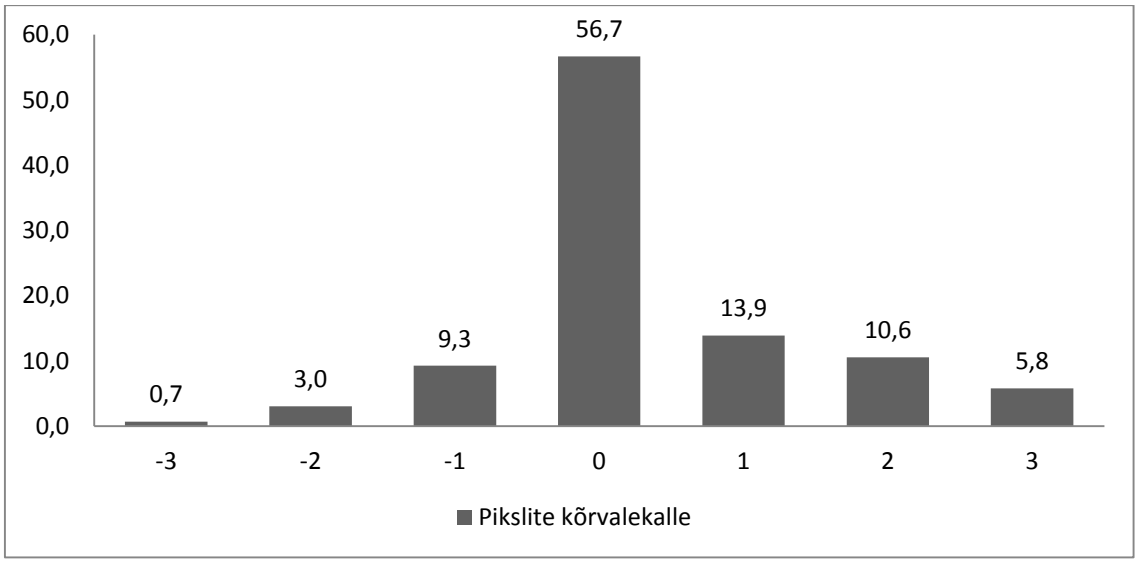
Esimene test:

- Kaugus – 1000[mm]
- Mõõtmiste arv – 1000
- Väljund – 1000 kaadrit üksteise otsas
- Teek – Kinect SDK



Joonis 11 Kinecti müra võrreldes objektiga

Salvestatud pildil (Joonis 11) on näha, kui palju pikslid kalduvad kõrvale objektist. Objektiks on sileda pinnaga ristkülik, mille äärejooned on valge ristkülikuga pildil märgistatud. Võimalik on näha minimaalset müra pildi peal ja alati on tegureid, mis mõjutavad sügavusandmete mõõtmisi.



Joonis 12 Pikslite kõrvalekalde histogramm

Histogrammile (Joonis 12) on kantud üles pikslite kõrvalekaldumised tegeliku objekti suurusest, ning millisest pikslist võeti sügavusandmed. Testi põhjal on oodata hinnanguliselt kolme piksli võrra kõrvalekallet nii horisontaal- kui ka vertikaalsuunas. Kaalutud keskmise arvutamisel saadi tulemuseks 0,35 ning, mis võib tähendada, et kaalutud keskmine on alam-piksel. Võimalik, et erinevused tulevad sügavuse arvutamisel, ja müra tekibki ikkagi objekti punktist kuni sensorjooneni. Kuid tänu mürale on võimalik leida alam-piksel.

4.2 Laserjoone leidmine

Laseriga kaardistamisel on kõige tähtsam osa laserjoone leidmisel. Peatükis käsitletakse optimaalsemaid mooduseid laserjoone leidmisel. Laserjoonega on võimalik märgata väiksemaid muutusi objektidel või pinnal. Silmaga on neid kerge eristada, kuid tähtis on saada arvuti seda tuvastama. Tuvastamine on piisavalt keeruline, sest väljakutseteks saab veel erinevate valgustusega keskkonnad, optilised mürad(optical flow [12]). Kõik need probleemid tuleb lahendada pilditöötlusel.

Pilditöötlus koosneb[13]:

- Pildi eeltöötlus
- Laserjoone eraldamine
- Tunnusjoonte eraldamine

Põhilised teemad, mida eeltöötluses käsitletakse on, valik 8-bitise või 12-bitise pildi vahel kuna XIMEA kaameral on võimalik salvestada kahes formaadis. Bayeri kodeering[14][15] või RGB väljundiga pilt, sest on võimalus kasutada punase laseri asemel ka rohelist. Tähtis on müra vähendamine kaadritel, millega vältida üleliigseid andmeid.

Laserjoone eraldamine hõlmab endas pildi tegemist kaheks erinevaks formaadis, kas RGB või HSV[16]. Pärast seda tuleb märkida laserjoon ka joonena saadud pildile.

Viimane etapp on tunnusjoonte eraldus ehk laserjoone leidmisel tuleb see ära märgistada, milliseid pikslid ära märgistada ja edasi töödelda.

4.2.1 Pildi eeltöötlus

Eelnevalt mainitult keskendub see osa pildi müra vähendamisele ning kvaliteedi parandamisele, mis aitaks paremini leida laserjoone asukohta.

Müra vähendamiseks on võimalik pilti siluda ja teha pilt loetavamaks protsessorile. Sellega vähendatakse keerulisi punkte, mis võivad anda valeandmeid, kui laserjoont leitakse.

Pildi kvaliteedi määramisel tuleb ta teisendada kas 8-bitiseks pildiks või 12-bitiseks, ja teha kindlaks millist formaati on mõistlik kasutada. Tuleb leida kui suur vahe on laserjoone leidmisel 12-biti ja 8-bitise pildi vahel.

Lühike peatükk Bayeri filtrist, mille piltidel on rohelisi piksleid kaks korda rohkem võrreldes punase ja sinisega. Kui salvestada pildid, kasutades rohelist laserit, Bayeri kodeerimisega koos, siis kas kvaliteet paraneb võrreldes punase laseriga.

4.2.1.1 Pildiformaat 8-bit või 12-bit

Alapeatüki põhiidee on uurida, milline formaatidest sobib kõige rohkem, salvestamaks laserjoone infot. Tuuakse välja erinevused 8-bitise ja 12-bitise värvimudeli vahel ning esitatakse kõige optimaalsem formaat, mida töös edasi kasutatakse.

8-bitine värvimudel[17][18] koosneb kolmest värvist: punane, roheline, sinine. Eksisteerib mudelis 256 kombinatsiooni ja varjundit iga värvi jaoks. 256 x 256 x 256

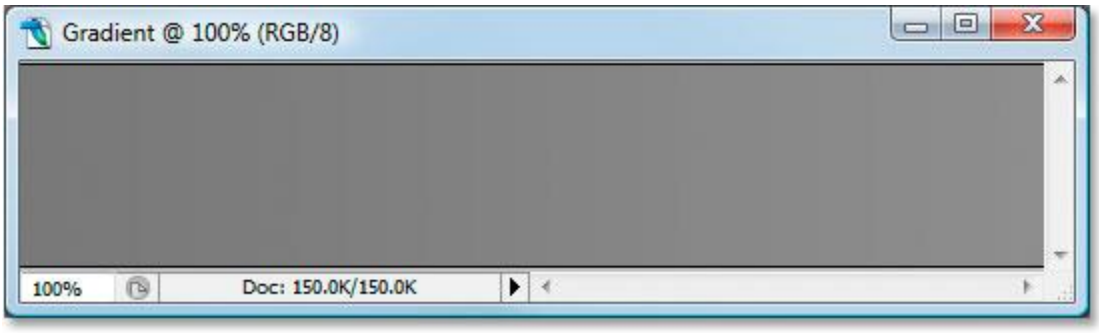
annab eksisteerivate värvide hulga 16,8 miljoni kanti. 16,8 miljonit värvi on piisav arv värve, et pildikvaliteet oleks rahuldav ning võimalik ka laserjoont leida.

Paljud pildid ei vajagi nii palju värve[19]. Kuid seal on ikkagi negatiivseid osasid 8-bitiste piltidega töödeldes ning vahel suurem on parem. Negatiivne külg on, kui töödelda 8-bitist pilti, siis tulevad välja pildikvaliteedi vähenemised. Kui võtta pilt (Joonis 13), mis on tavaline 8-bitine pilt ja pildi gradient(Joonis 14).

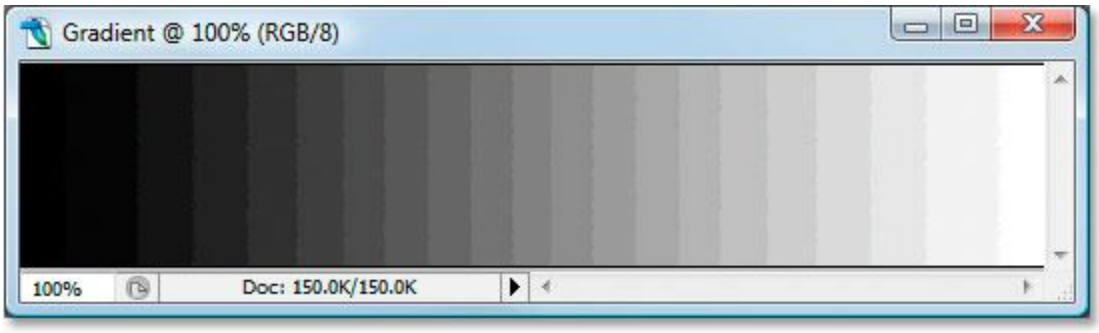
Joonis 13 8-bitine töötlemata pilt[19]

Joonis 14 8-bitise pildi muutusaste

Pilt on kvaliteetne ja tema värve on kerge näha ja mingeid kvaliteedimuutusi pole. Muutusaste on sujuv tumedast valgemaks. Töödeldes pildi muutusastet natuke ja surudes kogu musta ja valge värvi limiidid ühte alasse keskel, saab sellise muutusastme(Joonis 15). Peale töötlemist muudetakse musta ja valge värvi limiidid samaks, mis nad olid algselt (Joonis 16).



Joonis 15 Muutusaste peale töötlemist[19]



Joonis 16 Muutusaste peale töötlemist algstaadiumis[19]

Piltidelt on näha (Joonis 14;15;16), kuidas peale töötlemist muutus 8-bitise pildi muutusaste, see pole enam väga sujuv ning on näha, kus üks toon algab ja teine lõpeb. Järgmiseks näeme, mis on saanud 8-bitisest algpildist(Joonis 17).

Joonis 17 8-bitine pilt peale töötlemist[19]

Vesi pildil on justkui joonistatud ning on näha toonide erinevusi, kus üks lõpeb ja teine algab. Kvaliteet sai kannatada peale töötlemist. Kuid saab öelda, et 8-bitisel formaadil on ka positiivseid külgi[20].

Positiivsed küljed:

- Pildifailid on tunduvalt väiksemad
- Väiksem suurus tähendab kiiremat töötlust
- Paljude rakenduste jaoks on 8-bitine täiesti piisav

XIMEA kaameral on võimalik salvestada faile RAW-formaadis[21]. RAW-failid on minimaalselt töödeldud andmed otse pildisensorilt, mis hoiavad endas alles kõike infot pildi kohta näiteks teravustase, kontrast, küllastus, värvitemperatuur või valge tasakaal.

RAW-formaati salvestamisel on positiivsed küljed:

- Hoiab täpselt seda, mis jäädvustati
- Maksimaalne võimalik pildikvaliteet eraldamisel
- Seadeid võib muuta peale jäädvustamist, kaotamata kvaliteeti
- Võimalik töödelda efektiivsemalt arvuti peal
- Kui kasutada Bayeri filtrit on pildifail väiksem kui 8-bitine

RAW-formaati salvestamisel esineb negatiivseid külgi[22]:

- Suured failid (4-Megapiksline kaamera -> 4Mb fail)
- Töötlemine aeganõudev, nõuab iga jaoks manuaalset lähenemist
- Kindlat tarkvara on vaja nende töötlemiseks
- Nõuavad võimsamat seade töötlemiseks

Töötlemise kiiruse ja lihtsuse jaoks oleks mõistlik kasutada 8-bitist .jpeg formaati, mis võimaldaks andmeid kiiremini läbi lugeda, sest kaadreid on tuhandetes. Ning need kõik nõuavad töötlemist.

4.2.1.2 Bayeri kodeering või RGB väljund

Bayeri kodeering on tehnika, mis kannab igale pikslile väärtuseks punane, roheline või sinine. Värviandmed piksli kohta on võimalik tuletada kõrval olevate pikslite väärtusest.

Väljund näeb välja järgnevana:

1. RGRGRGR...
2. GBGBGBG...
3. RGRGRGR...
4. GBGBGBG...

Niimoodi on antud igale pikslile üks väärtus. On ka märgata, et rohelisi piksleid on kaks korda rohkem, kui teisi. Põhjuseks on, et inimsilm on tundlikum rohelisele, nii on võimalik info maksimaalne.

Kui tööks kasutada rohelist laserit ning pildile lisada Bayeri kodeering on suurem võimalus leida laseri piksleid kaadritel ja need sealt eraldada. See peab kahjuks jääma teemaks, mida saaks hiljem edasi uurida ja võimalus, mida kasutada. Ei olnud hetkel võimalusi testida rohelise laseriga. Jätkub töö punase laseriga ja tavalise RGB väljundiga.

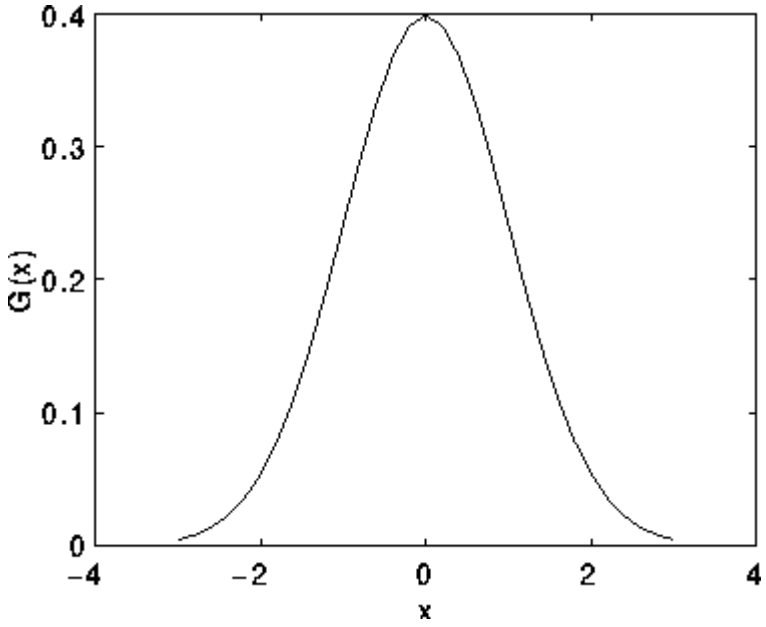
4.2.1.3 Gaussi silumine

Müra vähendamiseks on vaja pilti natuke eeltöödelda, selleks oleks vaja pilti hägustada. Pildi hägustamine likvideerib liigse müra ja detailsuse. Eeltöötlemine tekitab võimaluse eraldada rohkem värve ja leida laserjoon pildilt suurema tõenäosusega.

Selle jaoks on Gaussi silumine[23][24], mis just ’hägustabki’ pilte, et vähendada müra ning likvideerida detailsus. Gaussi jaotus on toodud valemina(1). Kus σ on jaotuse standardhälve.

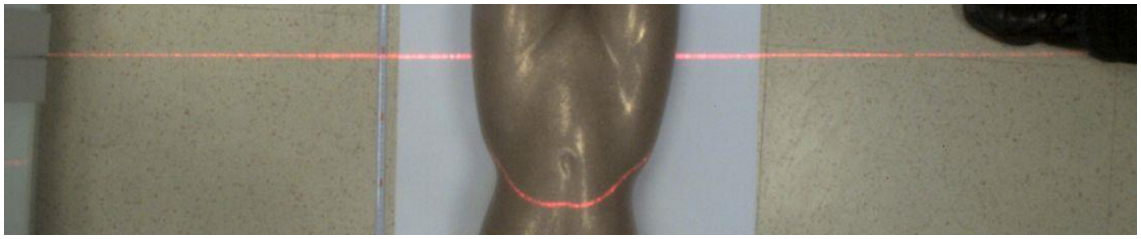
$$G(x) = \frac{1}{\sqrt{2\pi}*\sigma} * e^{-\frac{x^2}{2*\sigma^2}} \quad (1)$$

Gaussi jaotusel on kindel tuum(inglise keeles – kernel), mis esitab end Gaussi küüru kujul (’kella kujuga’). Kui võtta näiteks, mille jaotuse keskmine on 0 (x = 0). See on kujutatud graafikul (Joonis 18).



Joonis 18 Gaussi jaotus keskmisega 0 ja σ=1[24]

Järgmisel pildil on katse, mis on tehtud laserjoonega esimene pilt (Joonis 19).



Joonis 19 Esimene pilt 8-bitises formaadis

Joonis 20 Esimene pilt peale Gaussi silumist

Järgmine pilt on peale Gaussi silumist (Joonis 20). Näha on kuidas pilt läheb küll vähem detailsemaks, kuid müra on vähenenud. Laserjoon on ikkagi näha, sest ta on tugev punane ja on nähtav.

4.2.2 Laserjoone eraldamine

Peamine töö eraldada algpildist laserjoon, mida saab pärast manipuleerida joonte tekitamisel. Kui pilt on üleliigseid piksleid, müra või andmeid täis, siis algoritmid, mis peaks leidma jooni ei tööta korralikult ja on võimalus, et tekivad vale-positiivsed.

Esimene võimalus, mida uurida on HSV[25]. Pilt jagatakse kolmeks osaks, mis kirjeldavad värve.

Teine võimalus on kasutada kolme põhivärvi eraldamist pildis. Pilt jagatakse jällegi kolmeks osaks: punane, roheline, sinine.

Kui üks võimalustest on valitud, siis tuleb lisada lävi, mis elimineeriks kõik üleliigsed toonid ja jätaks ainult alles selle piirkonna, mida meil vaja.

4.2.2.1 Hue-Saturation-Value

Hue-Saturation-Value jagab pildi kolmeks osaks, mille järgi on lihtsam filtreerida laserjoon pildilt. Kolme põhiosa on: „hue“ – värvitoon, „saturation“ – halli osakaal pildis, „value“ – heledus.

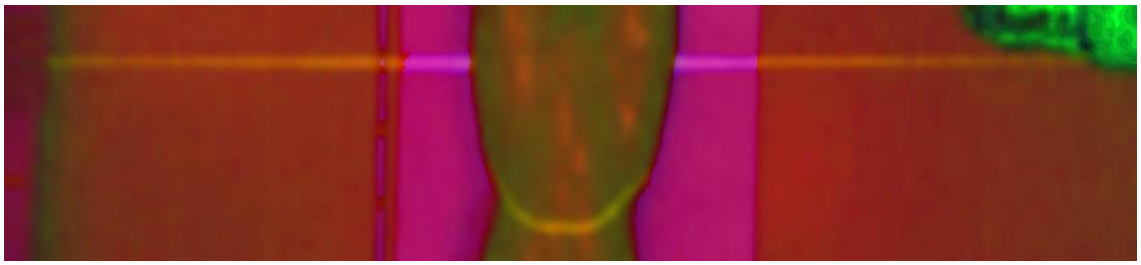
Kolme põhiosa esitusviis:

- Hue – Värvitoonid 0 kuni 360neni. Punane on 0-60
- Saturation – halli osakaal 0-100%
- Value – heleduse osakaal 0-100%

Selleks võtame pildi, mida on Gaussi poolt silutud(20) ja kasutame openCV-st saadud funktsiooni cvtColor(). Kogu kood on esitatud Lisa 1.

cvtColor(source, output, CV_BGR2HSV);

Funktsioon koos käsuga „CV_BGR2HSV“ teisendab tavalise kaadri HSV formaati. Ning tulemus näeb välja selline (Joonis 21). Pilt ei anna veel piisavat lihtsust.

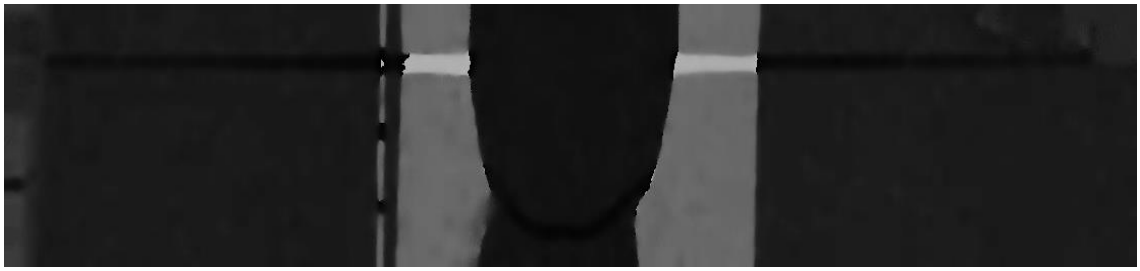


Joonis 21 Pilt RGB-st teisendatud HSV-ks

Lihtsus, mis on vajalik laserjoone eralduseks. Järgmisena tuleb pilt teha kolmeks, hetkel ta koosneb neist kolmest osast: „hue“, „saturation“, „value“. Kasutades lihtsat funktsiooni openCV-s split(). Kogu kood on esitatud Lisa 1.

split(source, hsvChannels);

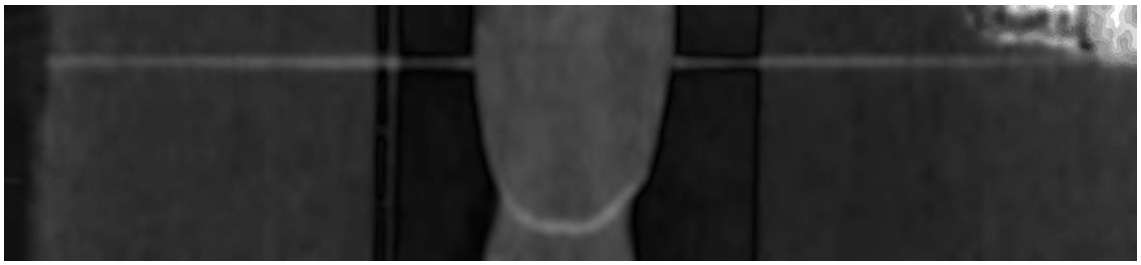
Funktsioon jagab pildi lahti ja saame esimese tulemuse „hue“ (Joonis 22).



Joonis 22 Hue kanal pildist

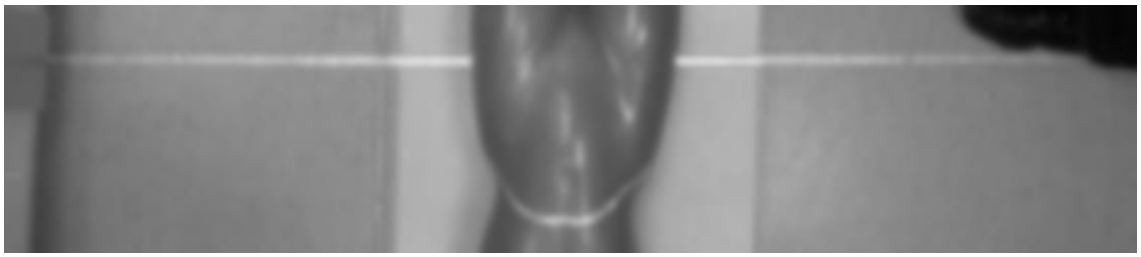
Selle peal on selgelt näha laserjoon, küll natuke tumedamal taustal, aga on olemas. Taust, milleks on valge paber muudab laserjoone väärtust heledamaks, kuid see ei tohiks mõjutada selle avastamist.

Teine tulemus on „saturation“ (Joonis 23). Näha on kuidas ülemine laserjoon on sama väärtusega, kui keha keskel, mis võimaldab valeandmete tekkimist. Võimalus on „saturation“ osa järgnevatest funktsioonidest välja jätta, see võib lõpptulemust segada. Hea on alles hoida igaks juhuks, kui peaks tekkima vale-positiive. Miks alles hoida, sest vahel võib teiste kanalite töötlemisel tekkida anomaaliaid ja saab kasutada „saturation“ kanalit, et anomaaliad likvideerida „and-lausega“.



Joonis 23 Saturation kanal pildist

Viimasena saame „value“ kanali pildi (Joonis 24). Kolmandal pildil on näha laserjoont, ehk siis mida heledam on osa pildist, seda rohkem on näha. Ning hiljem kasutada filtrit, mis eraldab kõige heledamad kohad oleks võimalik näha laserjoont.



Joonis 24 Value kanal pildist

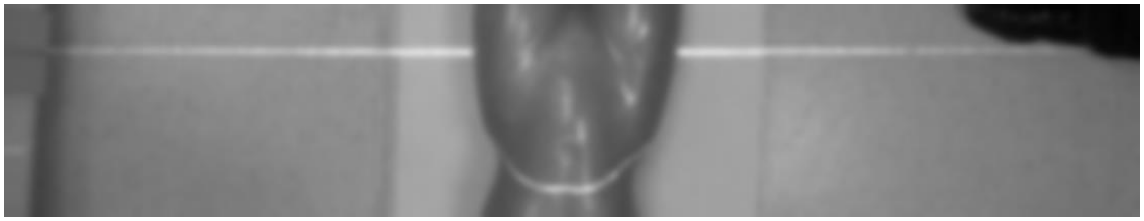
Uurimuse koha pealt on mõistlik jätkata „hue“ ja „value“ kanaliga piltidelt, sest need annavad kõige lihtsama informatsiooni, mis töötluseks vaja.

4.2.2.2 Red-Green-Blue

Tavapilt, mis on 8-bitises .jpg formaadis on juba RGB pilt ja koosneb kolmest kanalist, mida on kerge jaotada ning võrrelda.

Kasutades sama funktsiooni split(). Teeb kolm erinevat kanalit: punane, roheline ja sinine toon.

Esimene pilt on punase tooni kanal (Joonis 25). Mis eraldab punase tooni.



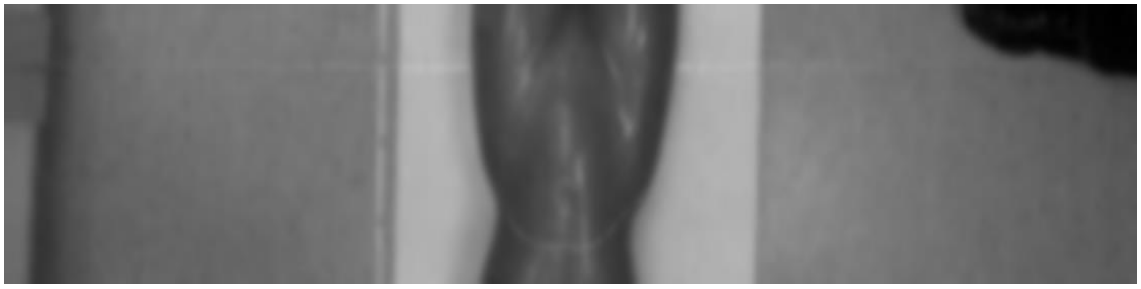
Joonis 25 Punane kanal

Toob tugevamalt esile kõige suurema punase tooniga pikslid pildilt. Ning punane laser on kergelt näha, kui kõige heledam osa pildist. Heledus tõuseb osaliselt valge paberiga ristumisel.

Teine pilt on rohelise tooni kanal (Joonis 26). Laser on täiesti kadunud ning rohelist tooni on peaaegu võimatu kasutada, kuid mõlemal pildil on objekti peal heledad punktid, mis võivad segama hakata. Saab kasutada rohelist kanalit, et viitena likvideerida need samad heledad punktid.

Joonis 26 Roheline kanal

Viimane pilt on sinise tooni kanal (Joonis 27). Täpselt sama juhtum, mis rohelise kanaliga. Ainuke erinevus on näha heleduse puudumises kindlates kohtades, mida kahjuks ei saa kasutada mitte millegi jaoks.



Joonis 27 Sinine kanal

Edasi tuleks kasutada punast kanalit maksimaalsemalt ja rohelist kanalit, viitepunktina mitmete vigade likvideerimiseks. Sinine kanal pole kasulik hetke oludes ning võib välja jätta kogu uurimusest.

4.2.2.3 Filtreerimine

Kui on olemas mõlema tehnoloogia nii RGB kui ka HSV pildid tuleb nendest välja filtreerida kõik üleliigsed väärtused: toonid, varjundid ja heledusastmed. Õige

seadistusega on lootus saada laserjoon eraldatud ning liikuda edasi objekti tunnusjoonte leidmiseni.

Selleks kasutame „thresholdingut“ (eesti keeles – lävendamine)[26]. *Thresholding* jagab pildi osadeks. Näiteks osad, mis jagavad sama väärtust on rohkem esile toodud, kui teised. Pildi objektid või detailid pannakse põhimõtteliselt taustale, millest kaotatakse ära suuremad detailid. *Threshold* töötab valemina (2), kus *T* on lävi ehk mis väärtus tuleb pikslil ületada või võrduda. Kui piksel täidab seda nõudmist, siis saab automaatselt maksimaalse väärtuse ja kui ei, siis minimaalse väärtuse.

$$g(x,y) = \begin{cases} 1 \; if \; f(x,y) \geq T \\ 0 \qquad otherwise \end{cases} \tag{2}$$

Esimeseks on lastud läbi „hue“ kanaliga pilt, millele on lisatud *thresholding* väärtus ja tulemus on pildil(Joonis 28). Pilt on väiksemaks tehtud *croppimise* käigus ning hetkel on näha laserjoont, mis on objekti keha alumisel osal.

Joonis 28 Hue pildi threshold

Teine on „value“ kanali pilt, kus on *thresholding* (Joonis 29). Näha on osaliselt laserjoont objekti peal, kuid mitte kõrvalaladel, sest valge taust on väga kõrge heledusega ja laser sulandub sinna. Kuid on võimalik kasutada *bitwise_and* lauses. Lause elimineerib kõik erinevused ja sarnased kohad jäävad alles.

Joonis 29 Value pildi threshold

Järgmisena liigume värvikanalite juurde ning rohelise kanali juures on *thresholding* lisatud, mis annab väljundiks (Joonis 30). Siin pole üldse midagi laserjoonest näha ning tuli idee kasutada HSV ja RGB eripärasi, et kokku panna pilt mõlemast.

Joonis 30 Rohelise kanali threshold

Lisades järgmiseks sinise kanali thresholdi (Joonis 31). Sinisel on täielikult tühi ala keskel. Ning mida rohkem katsetada, siis läheb pilt ainult segasemaks, sest sinisel kanalil polnud midagi laserjoonest näha.

Joonis 31 Sinise kanali threshold

Punase kanali thresholdiga(Joonis 32) tuli idee viia see kokku „hue“ kanali thresholdiga (Joonis 28). Selle jaoks on võimalik kasutada lauset *bitwise_or* mis võimaldab piltidel üheks saada oma eripärasustega.

Joonis 32 Punase kanali threshold

Joonis 33 Kombinatsioon Hue-st ja Punasest

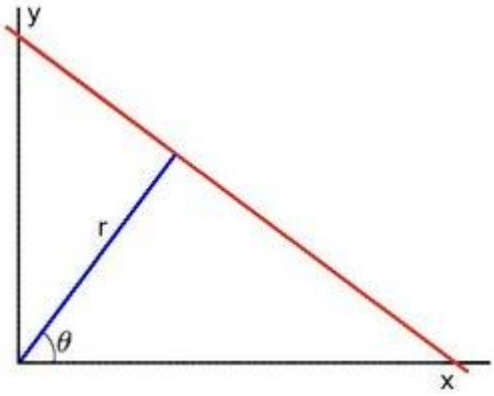
Pildil (Joonis 33) on näha, kui on pandud kokku „hue“ kanali ja punase kanali threshold. Ning hetkel on võimalik näha laserjoont seal.

Kõik on katsetuse küsimus, et leida sobivad *thresholdid* kõikidele kanalitele ning mõnedes valgustingimustega on võimalik ka ühe kanaliga läbi ajada. Väga kerge on katsetada staatilises keskkonnas, kuid kui keskkond on on kogu aeg muutuv, peab kood olema palju keerulisem.

4.2.3 Tunnusjoonte eraldus

Peatükk käsitleb, kuidas leida pildist laserjoon. Eesmärgiks on märgistada laserjoon pildil ja saada pikslite koordinaadid, mida hiljem kasutada 3-mõõtmelise ruumi koostamisel.

Laserjoone märgistuseks on kasutatud Hough teisendust[27]. Teisenduse eesmärgiks on leida piltidelt kindlaid tunnusjooni. Enamjaolt kasutatakse seda joonte, ringide või ellipsite tuvastamiseks. Tööpõhimõte on joonisel (Joonis 34). Kõige paremini kirjeldab seda valem(3).



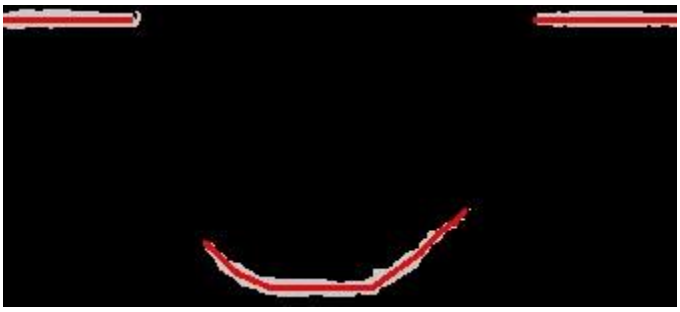
Joonis 34 Hough teisenduse tööpõhimõte[28]

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right) * x + \left(\frac{r}{\sin\theta}\right) \quad (3)$$

Põhimõtteliselt on iga r ja θ paaril olemas joon (x,y), mis neid läbib. OpenCV-s on olemas kasulik funktsioon, mis just leiabki jooned – HoughLinesP().

HoughLinesP(output, lines, rho, theta, threshold, minLineLength, maxLineGap);

Koodis[Lisa 1] kasutame seda funktsiooni viimase kanalite kombinatsiooni pildi(Joonis 33) peal ja saame tulemuse (Joonis 35). Laserjoon on märgistatud punase joonega ning joonel on pikslite väärtused (x,y) koordinaatide süsteemis.



Joonis 35 laserjoon märgistatud

5. Tulemuste rakendamine

Eesmärk on tulemuste põhjal leida sobiv rakendus, kuhu üks või teine seade sobib ning esitada 3-mõõtmeline kuju tulemustest.

Esiteks, kas Kinectiga oleks võimalik kaardistada maanteid või patsienti positsioneerida, teades testide tulemusi, mis Kinectiga tehtud on.

Teiseks, kas on võimalik kaardistada maanteid või patsienti positsioneerida XIMEA kaamera ja laserjoonega, kasutades eelmise peatüki uurimuse tulemusi.

5.1 Kinecti sobivus

On vaja kindlaks teha, mis rakenduste jaoks on Kinect kõige optimaalsem. Katsed Kinectiga andsid tulemuseks lahutusvõime 1[mm] või vähem teistel vahemaadel kui 1000[mm]. Optimaalne kaugus kõige paremate tulemustega on 700[mm] ja 1000[mm] vahel ning see sõltub inimese eelistusest ja kui suurt ala on vaja kaardistada.

On teada, et maantee kaardistamisel on väga tähtis sügavusandmete lahutusvõime, sest põhjuseks on erinevad moonutused pinnase kvaliteedis ning, mida täpsem on seade, seda parem on lugeda andmeid.

Teiseks probleemiks on kiirus, ning maanteel on vaja väga head põhjust aeglase kiirusega liikumisel. Ning mida kiiremini liigub auto, seda kiiremini saab lõpule viidud ka kogu protsess.

Kolmas probleem sõltub ilmastikuoludest nagu päikesevalguse intensiivsusest, kellaajast ja kas hiljuti on vihma sadanud. Erinevatel kellaaegadel on probleem valguses, mis segab Kinecti toimimist. Eriti kui Päike paistab ka tee peale, siis Kinect ei suuda anda täpseid andmeid. Sama juhtub ka lompide olemasolul peale vihma, sest sealt peegeldub valgus tagasi ja segab andmete töötlemist.

Oletame, et on auto sõidukiirusega 90[km/h] , mis on teisendades 25[m/s]. Kinecti kaamera kaadrite sagedus on 30[Hz] ehk 30[kaadrit/sekundis]. Jagades omavahel(4).

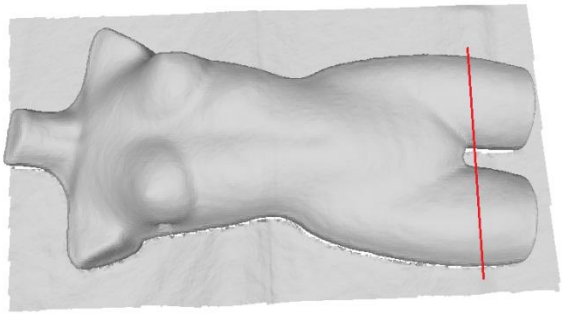
$$\frac{sagedus}{kiirus} = \frac{30\frac{kaadrit}{s}}{25\frac{m}{s}} = 1{,}2\ kaadrit/meetris \quad (4)$$

1,2 kaadrit/meetri kohta pole kõige parem tulemus tee kaardistamisel. Selle kaadrite hulgaga ei saa mitte mingit vajalikku infot salvestada, eriti veel kuna Kinectil on teatud viide andmete saamisel.

Patsiendi positsioneerimisel on tähtis täpsus ning lahutusvõime. Kui lahutusvõime oleks rohkem kui 1[mm], siis poleks seadmest kasu. Testi põhjal teame, et lahutusvõime on 1[mm] või vähem.

Positsioneerimisel on võimalik luua kontrollitud keskkond, kus valgustingimused on ideaalsed Kinecti jaoks. Teiseks on, et Kinecti kiirust või patsiendi „kiirust“ saab reguleerida, nii et saaks piisavalt täpsed tulemused ajaga.

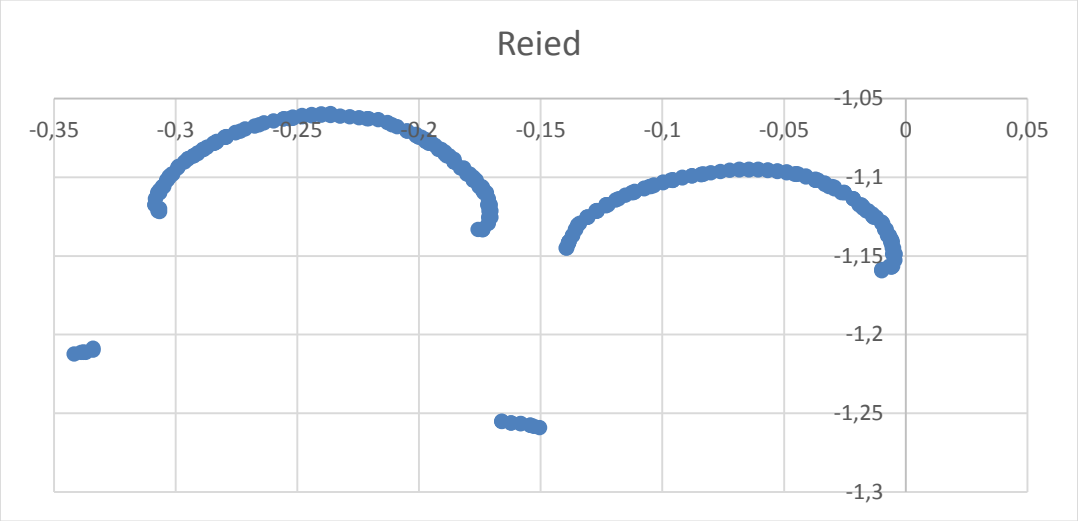
Kasutades Kinecti SDK funktsiooni Fusion on võimalik luua 3-mõõtmeline mudel „patsiendist“(Joonis 36). Mudelit saab salvestada .obj, .ply või .stl kujul, mida siis vastavad 3-mõõtmelise objektide manipuleerimistarkvaraga saab vaadelda. Probleemiks on, et salvestatud kuju ei sisalda *point-cloud* andmeid. Kuid algandmed enne salvestamist on 3-mõõtmelisel kujul ning on võimalik eraldada (x,y,z) koordinaadid. Märgistasin lõigu, kust on andmed võetud.



Joonis 36 Kinect fusion patsiendiga

Selleks ongi tehtud väike muudatus koodi, et oleks võimalik hetkeolude koordinaadid salvestada ning nende väärtused on esitatud siin joonisel meetrites(Joonis 37). Probleem

võib seisneda selles, et neid andmeid on tohutult palju ühe pildi kohta, mis võib tarkvara tegemisel osutuda andmete töötlemisel nõudvaks.



Joonis 37 Graafik andmete põhjal

5.2 Laserjoone sobivus

Peatükis loeteldakse laserjoone seadme sobivusi erinevate rakenduste jaoks. Katsed andsid tulemuseks, et on võimalik leida laserjoon pildilt ühte moodust kasutades.

Mooduseid on veel mitmeid, kuidas lõpuks leida laserjoon pildilt. Näiteks saab kasutada Gaussi funktsiooni teist tuletist[29] või mitmelaserilist süsteemi[13].

Katsed põhinesid ainult objekti peal laserjoone leidmisega, milleks oli mannekeen. Patsiendi positsioneerimises oleks väga hea kasutada laserjoone andmeid, mis kaamera suudab salvestada oma kaadritesse. Kuna kaamera resolutsioon on veel 2048x2048, siis on pilt väga detailne ja on võimalik saada kasulikke andmeid töötluseks. Laserjoon teeb objekti mitmeks lõiguks, mida on kergem pärast võrrelda uute kaardistamiskatsetega. Kuna on vaja teada, kas patsient on muutnud oma asendit kasvõi natukese võrra.

Andmed tuleb teisendada pildil (Joonis 35) 3-mõõtmeliseks. (x,y) koordinaadid tuleb teha (X,Y,Z) koordinaatideks, mida saab hiljem kanda graafikul tuues välja objekti eripärasused.

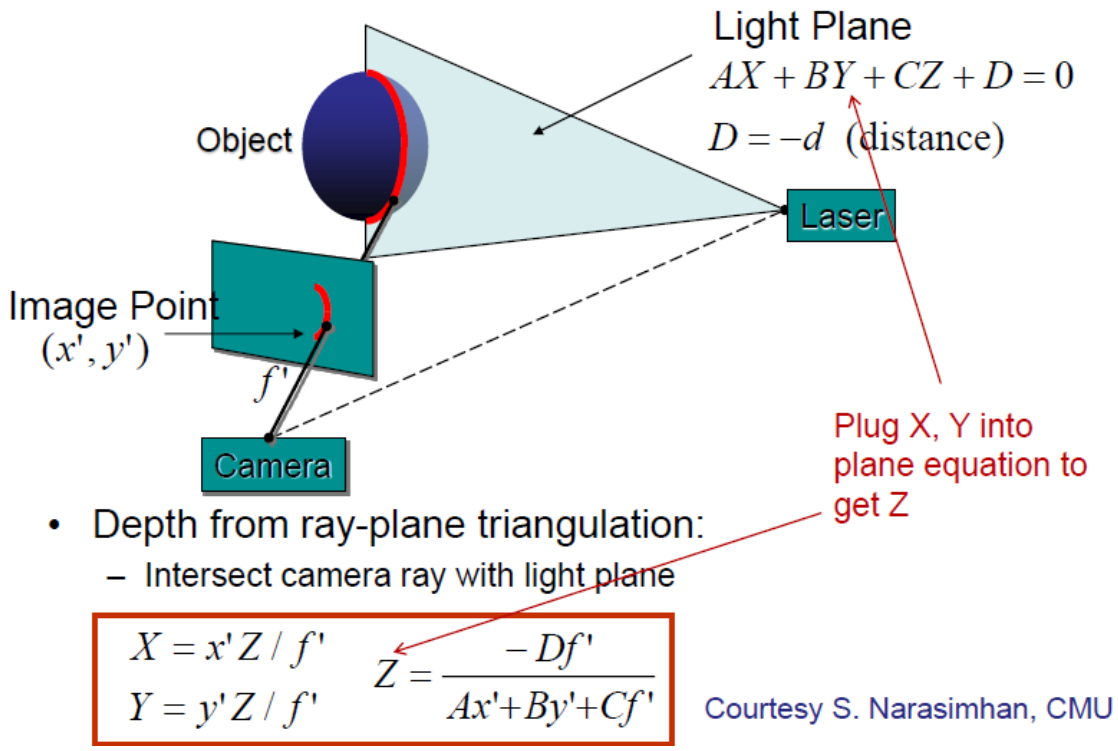
Selle jaoks on vaja teha kahte asja, kalibreerida kaamera ning triangulatsioon. Kaamera kalibreerimiseks on valem(5)[30], mille käigus leitakse fookuskaugused ja kaadri keskpunktid.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r11 & r12 & r13 & t1 \\ r21 & r22 & r23 & t2 \\ r31 & r32 & r33 & t3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (5)$$

Katsete käigus saime teada väärtused:

- fx – 1116,3
- fy – 1119,5
- cx – 1019,5
- cy – 1019,5

Edasi arvutatakse välja (X,Y,Z) koordinaadid, mille jaoks on valemid(Joonis 38)[31]. Niimoodi on võimalik kujutada pilt 3-mõõtmelises ruumis ja graafikul uurida.



Joonis 38 Triangulatsiooni valem[31]

Tegin viimaste katsete tulemusel testi pildiga (Joonis 39) ning, siis lisasin laserjoone eraldamise väljundi (Joonis 40). Arvutasin sinna otsa laserjoone kuju (Joonis 41) ja

lisasin selle originaalpildile (Joonis 42). Pildilt on näha, et kogu laserjoone pikkust pole alati vaja ning teatud alad võib välja filtreerida.

Arvutatud väärtused on toodud välja siin graafikul millimeetrites(Joonis 43). Anomaalia tekib, et väärtused vaikselt kogu aeg tõusevad või võib olla probleem lihtsalt asetuses. Hetkel ta tundub nagu oleks ristlõige laserjoonega märgistatud kohast.

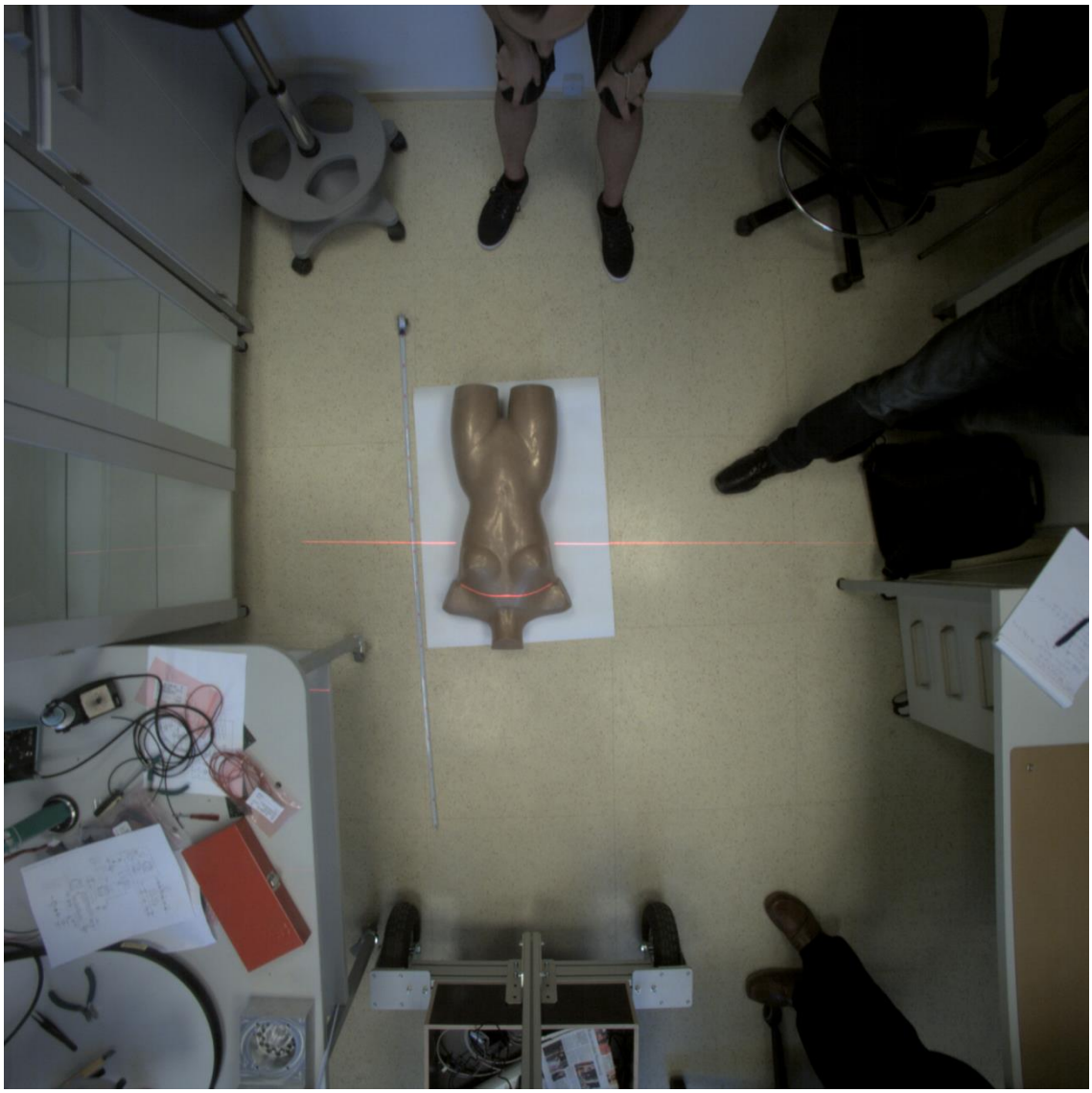
Tee kaardistamist testida ei saanud, kuid teiste uurimuse käigus on see mõeldav. Kui võtta et laserjoont on võimalik leida tavalise objekti peal.

Probleeme tekitavad ilmastikuolud nagu ere valgus ja lombid tees. Ere valgus võib täielikult kaotada ära laserjoone. Veelombid neelavad endasse laserjoone või siis peegeldavad tagasi raskendades andmete loetavust.

Probleeme ei tohiks tekitada enam kiirus, sest XIMEA kaamera kaadrite sagedus on 90[Hz] ja kasutades valemit(6).

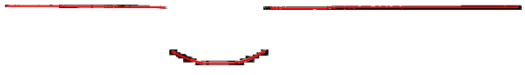
$$\frac{sagedus}{kiirus} = \frac{90\frac{kaadrit}{s}}{25\frac{m}{s}} = 3{,}6\ kaadrit/meetris \qquad (6)$$

3,6 kaadrit/meetri kohta pole halb, ning annab võimaluse teed kaardistada ka laserjoone seadmel.

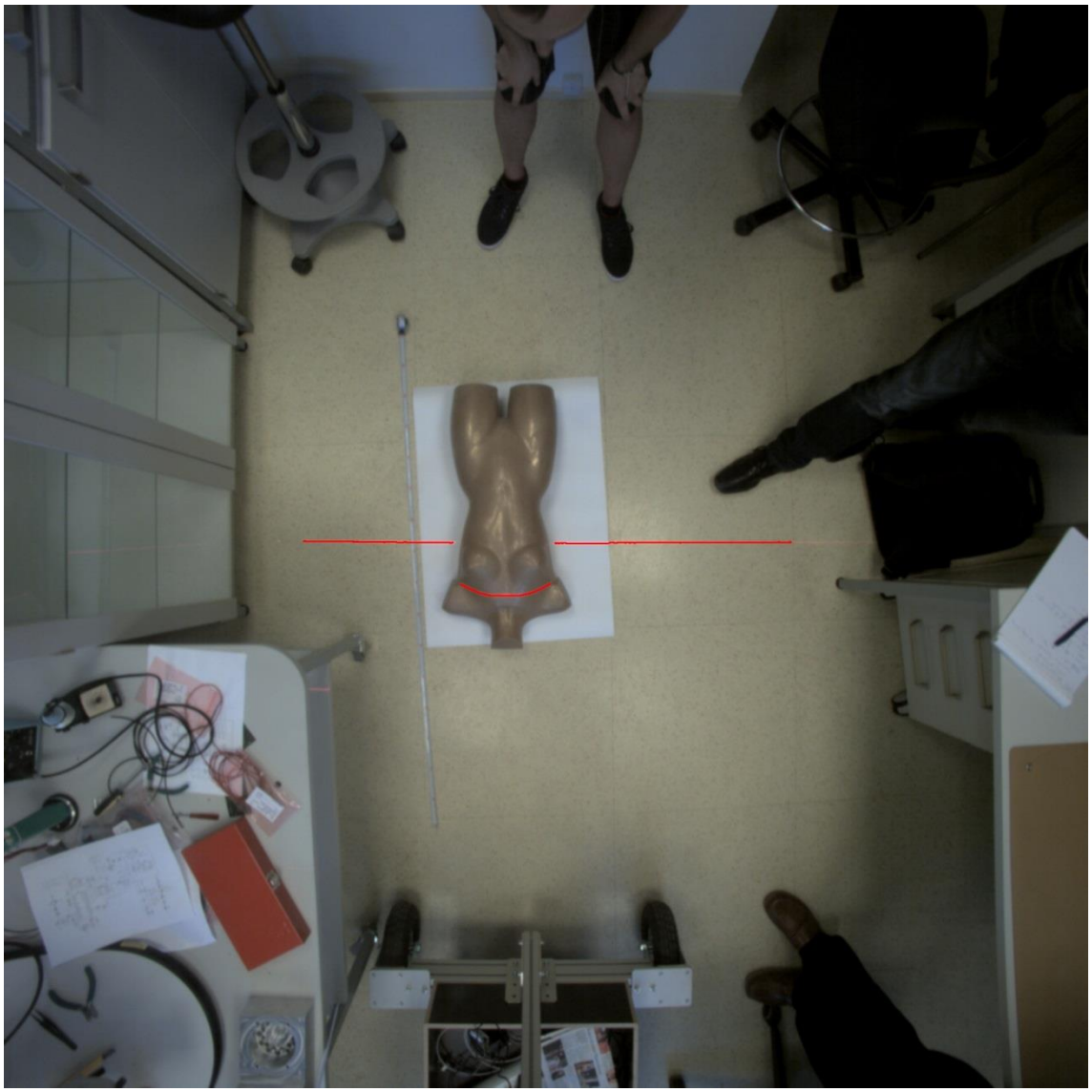


Joonis 39 Testi algpilt

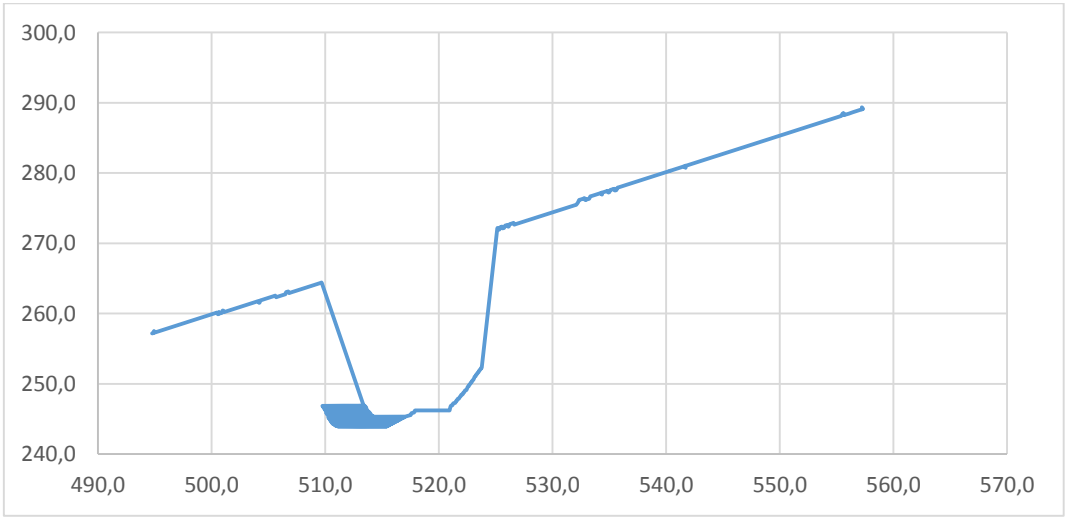
Joonis 40 Laserjoone väljund peale filtreid



Joonis 41 Laserjoone kuju peale joonarvutust



Joonis 42 Algpildile lisatud arvutatud laserjoon



Joonis 43 Graafik pärast tulemuste arvutamist

6. Kokkuvõte

Eesmärgiks oli testida kahe laserseadme, Kinect ja laserjoone kiirgaja koos kaameraga, sobivust kahe rakenduse jaoks. Üks rakendus oli patsiendi positsioneerimine – leida patsiendi asend ja kujutada see graafiliselt, et võrrelda uuel korral. Teine rakendus oli teeolude kaardistamine, kui seade on kinnitatud sõidukile ja liikumise pealt kaardistada tee seisukord.

Testid seisnesid selles, et teha kindlaks kas seadmega oleks võimalik midagi taolist teha rakenduste puhul ning saada kinnitust, et on võimalik luua reaalsem tarkvara.

Kinecti testidega sai kindlaks, et on piisavalt täpne ja lahutusvõime langeb 1[mm] ja alla alasse. Kõik oleneb kaugusest, ning kõige optimaalsem kaugus 1[mm] täpsuse jaoks on 700[mm] – 1000[mm] piirkonnas. Kasutusel oli Kinect Fusion, millega loodi 3-mõõtmeline pilt objektist ja eraldati koordinaadid ühe lõigu kohta. Koordinaatidest loodi ristlõige ja see vastas ristlõikele, mis oli märgitud objekti peal.

Laserjoone seadmega on võimalik leida laserjoon. Peale piisavat silumist, on müra vähendatud tasemeni, kus saab pilti kergemini töödelda. Sai kasutatud erinevaid värvitoonide kanaleid nagu RGB ja HSV, et luua lõplik pilt, kus ainult laserjoon on peal. Selle jaoks tuleb alati teada keskkonna valgusnäitajaid ja kontrollitud keskkond on kõige parem selle jaoks. Laserjoone pikslite koordinaatide põhjal oli võimalik leida Z-koordinaat andes võimaluse kujutada objekti ristlõiget, kuid graafikul oli näha .

Teeolude mõõtmist ei olnud võimalik katsetada, aga testide põhjal võib eeldada. Kinect ei sobi teeolude mõõtmise jaoks kuna ta kaamera pole piisavalt võimas, et rahuldavaid kaadreid tekitada ja viiteaeg võib rikkuda andmete esituse. Täpsus on testitud ja sobilik 1[mm], aga see on ainuke positiivne punkt siin rakenduses kuna Kinecti mõõtmisi segab ere valgus. Laserjoone seadmega on võimalused paremad, sest kaamera on võimas ja suudaks parimaid kaadreid pakkuda kahest. Kõige suurem probleem seisnebki keskkonnas, sest see pole kontrollitud keskkond ja ilmastikuolud suudavad tugevalt mõjutada mõõtetulemusi.

Patsiendi positsioneerimisel oli objektiks mannekeen, mille peal katsetati Kinecti võimet jäädvustada 3-mõõtmelist pilti ning esitusandmeid lugeda graafikul. Kinecti andmete põhjal loodud graafik oli rahuldav ning näitab ka võimalust luua palju parem

tarkvara, mis toetaks kahe erineva pildi võrdlemist, sest seda toetab täpsus 1[mm]. Lasersjoone seadmel on võimalik leida kiir iga kaadri pealt ja koordinaatide põhjal luua Z-koordinaat, mis aitaks lõpuks võrrelda erinevaid asendeid patsiendi peal, sest andmed on olemas. Patsiendi positsioneerimisel on soodustavaks faktoriks kontrollitud keskkond ja suurem osa müraallikatest on võimalik likvideerida, tehes tarkvara loomise väga lihtsaks.

Uurimuse põhjal soovitaks Kinecti patsiendi positsioneerimise jaoks, sest olemasolev tarkvara võimaldaks alustada väga soodsast piirkonnast. Senikaua kuni Kinect asetseb optimaalses kauguses on parimad täpsusnäitajad garanteeritud. Luues võimaluse andmeid rahuldavalt võrrelda.

Kogutud andmetega on võimalik edasi liikuda luues parem tarkvara Kinecti jaoks, mis aitaks täpsemini salvestada patsiendi koordinaadid ja neid lihtsamini töödelda. Alustustöö on tehtud suurelt osalt Kinect Fusioni poolt. Laserjoone leidmisel tuleks edasi liikuda ja töödelda mitut laserjoone asetust korraga, et oleks võimalik luua 3-mõõtmeline mudel objektist.

Kasutatud Kirjandus

[1] *Kinect for Windows Sensor Components and Specifications*

http://msdn.microsoft.com/en-us/library/jj131033.aspx (30.05.2014)

[2] *Ultra-compact USB3 Vision industrial cameras*

http://www.ximea.com/en/products-news/usb3-industrial-camera (30.05.2014)

[3] *Skeletal Tracking*

http://msdn.microsoft.com/en-us/library/hh973074.aspx (30.05.2014)

[4] *Depth Stream*

http://msdn.microsoft.com/en-us/library/jj131028.aspx (30.05.2014)

[5] **M. R. Andersen, T. Jensen, P. Lisouski, A.K.Mortensen, M.K. Hansen,T.Gregersen and P. Ahrendt,** *Kinect Depth Sensor Evaluation for Computer Vision Applications* http://eng.au.dk/fileadmin/DJF/ENG/PDF-filer/Tekniske_rapporter/Technical_Report_ECE-TR-6-samlet.pdf (30.05.2014)

[6] *Kinect Constants*

http://msdn.microsoft.com/en-us/library/hh855368 (30.05.2014)

[7] *MQ042CG-CM Overview*

http://www.ximea.com/en/products/usb3-vision-cameras-xiq-line/mq042cg-cm (30.05.2014)

[8] *Kinect for Windows Programming Guide*

http://msdn.microsoft.com/en-us/library/hh855348.aspx (30.05.2014)

[9] *Introduction to OpenCV*

http://docs.opencv.org/modules/core/doc/intro.html (30.05.2014)

[10] *Getting started with Visual Studio*

http://www.visualstudio.com/get-started/overview-of-get-started-tasks-vs (30.05.2014)

[11] *What is R*

http://www.inside-r.org/what-is-r (30.05.2014)

[12] *Optic Flow*

http://www.scholarpedia.org/article/Optic_flow (30.05.2014)

[13] **K.Sung, H.Lee, Y.S. Choi, and S.Rhee,** *Development of a Multiline Laser Vision Sensor for Joint Tracking in Welding*

[14] *Bayer Encoding for Color Images*

http://www.ni.com/white-paper/3903/en/ (30.05.2014)

[15] *RGB "Bayer" Color and MicroLenses*

http://www.siliconimaging.com/RGB%20Bayer.htm (30.05.2014)

[16] *Color Principles - Hue, Saturation, and Value*

http://www.ncsu.edu/scivis/lessons/colormodels/color_models2.html (30.05.2014)

[17] *8-bit Color Images*

http://homepages.inf.ed.ac.uk/rbf/HIPR2/8bitcol.htm (30.05.2014)

[18] **Laura Shoe**, *8 bit, 12 bit, 14 bit, 16 bit — What Does It Really Mean to Digital Photographers?* published 09.08.2011

http://laurashoe.com/2011/08/09/8-versus-16-bit-what-does-it-really-mean/ (30.05.2014)

[19] **Steve Patterson** *The Benefits Of Working With 16-Bit Images In Photoshop*

http://www.photoshopessentials.com/essentials/16-bit/ (30.05.2014)

[20] *Understanding Raw Files*

http://www.luminous-landscape.com/tutorials/understanding-series/u-raw-files.shtml (30.05.2014)

[21] **Bob Atkins.** *RAW, JPEG and TIFF*, 2004 (updated June 2008)

http://photo.net/learn/raw/ (30.05.2014)

[22] *RAW FILE FORMAT*

http://www.cambridgeincolour.com/tutorials/RAW-file-format.htm (30.05.2014)

[23] *Gaussian filter, or Gaussian blur*

http://www.librow.com/articles/article-9 (30.05.2014)

[24] *Gaussian Smoothing*

http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm (30.05.2014)

[25] *Hue, Saturation & Value*

http://www.greatreality.com/color/ColorHVC.htm (30.05.2014)

[26] **Bryan S. Morse** *Lecture 4: Thresholding*, 12.01.2000

http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/threshold.pdf (30.05.2014)

[27] *Hough Transform*

http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm (30.05.2014)

[28] *Hough Line Transform*

http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html (30.05.2014)

[29] **V.Matiukas, D. Miniotas** *Detection of Laser Beam's Center-Line in 2D images*, published 2009

[30] *Camera Calibration and 3D Reconstruction*

http://docs.opencv.org/2.4.5/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=fundamental#cv2.findFundamentalMat (30.05.2014)

[31] **Guido Gerig,** *Structured Lighting*. CS 6320, 3D Computer Vision

http://www.sci.utah.edu/~gerig/CS6320-S2012/Materials/CS6320-CV-S2012-StructuredLight.pdf (30.05.2014)

[32] **Z-Laser ZM18 Series**

ftp://ftp.stemmer-imaging.com/websites/documents/products/illumination/Z-Laser/en-Z-Laser-ZM18-BZLAS3-201402.pdf (02.06.2014)

Lisa 1 – Laserjoone töötlemise kood

```
#include "stdafx.h"
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/core/core.hpp"
#include <iostream>
#include <fstream>
#include <algorithm>

using namespace cv;

Mat remote;
Mat source;
int threshold_value = 0;
int threshold_type = 0;;
int const max_value = 255;
int const max_type = 4;
int const max_BINARY_value = 255;

int main()
{
	float A = -0.00025;
	float B = 0.001949;
	float C = 0.000407;
	float D = -1.0;
	float cx = 319.5;
	float cy = 239.5;
	float fx = 525.0;
	float fy = 525.0;
	short hue_min = 139;
	short hue_max = 200;
	short sat_min = 80;
	short sat_max = 200;
	short val_min = 200;
	short val_max = 200;

	short blu_min = 140;
	short blu_max = 200;
```

```
	short grn_min = 237;
	short grn_max = 200;
	short red_min = 122;
	short red_max = 200;

	int MAX_KERNEL_LENGTH = 7;

	Mat img =
imread("C:/Users/eekasoo/Desktop/Lõputöö/Pildid/UUS/Orig2.bmp");
	Mat edge2;
	Mat grayImg, hsvImg;
	imshow("original", img);
	for (int i = 1; i < MAX_KERNEL_LENGTH; i = i + 2)
	{
		GaussianBlur(img, img, Size(i, i), 0, 0);
	}

	//edge2 = edge(img);
	imshow("Gaussian_1", img);

	cvtColor(img, grayImg, CV_BGR2GRAY);
	cvtColor(img, hsvImg, CV_BGR2HSV);

	Mat dst, cdst;
	vector<Mat> channels;
	vector<Mat> newChannels (3);
	vector<Mat> colourChannels(3);
	split(hsvImg, channels);
	split(img, colourChannels);

	threshold(channels[0], channels[0], hue_min, hue_max,
THRESH_BINARY);
	threshold(channels[1], channels[1], sat_min, sat_max,
THRESH_BINARY);
	threshold(channels[2], channels[2], val_min, val_max,
THRESH_BINARY);

	//while (true)
	//{
	//	int c;
	//	c = waitKey(20);
	//	if ((char)c == 27)
```

```
//    {
//          break;
//    }
//}

      threshold(colourChannels[2], newChannels[2], red_min, red_max,
THRESH_BINARY);
      threshold(colourChannels[2], colourChannels[2], 209, red_max,
THRESH_BINARY);
      threshold(colourChannels[1], colourChannels[1], grn_min,
grn_max, THRESH_BINARY);
      threshold(colourChannels[0], colourChannels[0], blu_min,
blu_max, THRESH_BINARY);

      bitwise_and(colourChannels[2], channels[0], newChannels[0]);
      bitwise_and(newChannels[2], channels[1], newChannels[1]);
      bitwise_xor(newChannels[0], newChannels[1], newChannels[1]);

      namedWindow("REDSAT", WINDOW_NORMAL);

      imshow("REDSAT", newChannels[1]);

      cvtColor(newChannels[1], cdst, CV_GRAY2BGR);

      vector<Vec4i> lines;
      HoughLinesP(newChannels[1], lines, 1, CV_PI / 180 * 180 , 1, 1,
25); //*70 diagonal, 110 diagonal, 150 vertical

            for (size_t i = 0; i < lines.size(); i++)
      {
            line(img, Point(lines[i][0], lines[i][1]),
Point(lines[i][2], lines[i][3]), Scalar(0, 0, 255), 1, 8);
      }
      for (size_t i = 0; i < lines.size(); i++)
      {
            line(cdst, Point(lines[i][0], lines[i][1]),
Point(lines[i][2], lines[i][3]), Scalar(0, 0, 255), 1, 8);
      }

      std::ofstream myfiles;
      myfiles.open("C:/Users/eekasoo/Documents/XYZ.txt");
      for (size_t i = 0; i < 796; i++)
      {
```

```
            float Z;
            float X;
            float Y;
            float x = lines[i][0];
            float y = (lines[i][1] + lines[i][3]) / 2;
            Z = (-D*fx) / (A*x + B*y + C*fx);
            X = (x * Z) / fx;
            Y = (y * Z) / fx;
            std::cout << X << "\t" << Y << "\t" << Z << std::endl;
            myfiles << X << "\t" << Y << "\t" << Z << "\n";
      }
      myfiles.close();

      namedWindow("Original image with Lines", WINDOW_NORMAL);
      namedWindow("Laser lines", WINDOW_NORMAL);
      imshow("Original image with Lines", img);
      imshow("Laser lines", cdst);

      waitKey();
}
```