TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Informaatika eriala

Janno Jõgeva

Hüperboloidpeegliga roboti kaamera pildi kalibreerimine

Bakalaureusetöö (6 EAP)

Juhendaja: Sven Laur

Autor: .....„.....“ mai 2012

Juhendaja: .....„.....“ mai 2012

Lubada kaitsmisele
Professor: .....„.....“ mai 2012

Tartu 2012

Sisukord

Sissejuhatus

Töö teema tuleneb praktilisest vajadusest Robotex-i robootikavõistluse ülesande täitmisel ja isiklikust huvist selliste programmide toimimise vastu. Robootikas ja ka mõnes teises valdkonnas, nt. turvakaamerate puhul, kasutatakse väga laia vaatenurgaga süsteeme. Laia vaatenurka on vaja selleks, et saada võimalikult palju informatsiooni ümbritseva keskkonna kohta. Täpsem ülesande kirjeldus on esitatud esimeses peatükis. Laia vaatenurga saavutamiseks on mitmeid võimalusi. Neid võimalusi vaadatakse täpsemalt teises peatükis.

Robootikas on tihti oluline teada kaugust konkreetsete objektideni. Näiteks on vaja hinnata liikumistrajektooril asuvate takistuste kaugust ja mõõtmeid. Sellise teadmise saavutamiseks tuleb tekitada seos kaamerast tuleva pildi ja päris maailma vahel. Esimeses lähenduses piisab tihti seosest reaalse maailmaga roboti liikumisega samas tasapinnas. Kui ratastega robot sõidab põrandal, piisab põrandal asuvate takistuste kauguse teadmisest. Kolmandas peatükis esitatakse valik olemasolevatest programmidest, mis on mõeldud kaamera kalibreerimise ülesande täitmiseks. Neljandas peatükis on toodud viis testandmete genereerimiseks ja põhjendatakse testandmete kasutamist. Viies peatükk käsitleb kalibratsioonimustri tuvastamist. Kuuendas peatükis vaadatakse üht võimalikku viisi tuvastatud kalibratsioonimustritest saadud info kasutamiseks kaamera kalibreerimisel.

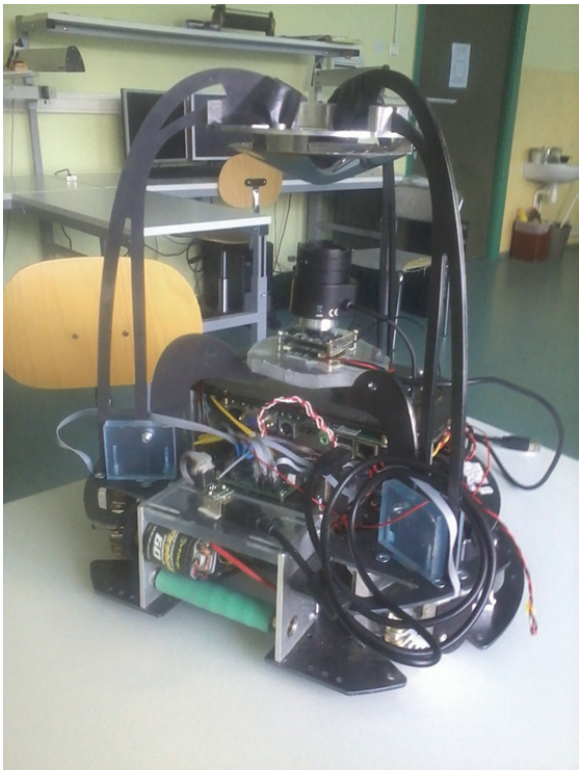
Bakalaureusetöö eesmärk on uurida võimalusi sellise programmi kirjutamiseks, mis on võimeline genereeritud piltide põhjal esitama teisenduse pildi ja mingi kindla reaalse maailma tasandi vahel - näiteks roboti all olev põrand. Lisaks soovitakse õppida POVRay tarkvarapaketi kasutusvõimalusi testandmete genereerimisel ja tutvuda olemasolevate tarkvaraliste lahendustega.

Ingliskeelses kirjanduses kasutatakse mõistet *camera calibration* üldmõistena igasuguse kaamera kalibreerimise kohta. Töös räägitakse kalibreerimisest pildil asuva tasandi „venituse“ mõttes. Kalibreerimise all ei mõelda värviruumiga vastavusse panemist (*photometric camera calibration)*, kui ei ole märgitud teisiti.

Töö lisades on toodud näide testpiltide genereerimiseks vajalikest konfiguratsioonifailidest. Samuti on esitatud link Python-is kirjutatud programmile, mis otsib genereeritud piltidelt kalibratsioonimustrit.

1 Ülesande kirjeldus

Robotex on Eestis korraldatav robootikavõistlus, kus kolme viimase aasta põhiülesandeks on olnud lihtsustatud jalgpall. Seda ülesannet lahendavad korraga kaks robotit, kes üritavad mängu jooksul võimalikult palju palle vastasroboti väravasse saata. Joonisel 1 on näha üht selle ülesande tarbeks ehitatud robotit. Ülesanne on suures osas inspireeritud rahvusvahelisest võistlusest *Robocup*, mis on robotijalgpalli mõistes maailmameistrivõistlused.



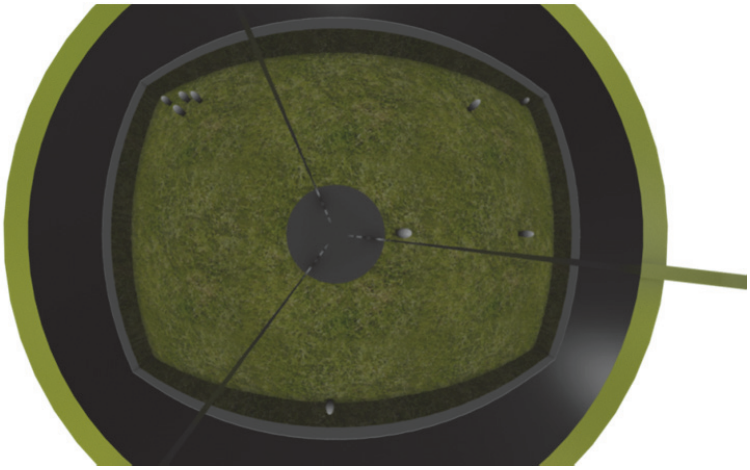
Joonis 1 Ehitusjärgus olev robot. Pildi keskel on näha objektiiv (must element pildi keskel) ja selle kohal paiknev hüperboloidpinnaga peegel.

1.1 Robotex-i keskkonna kirjeldus

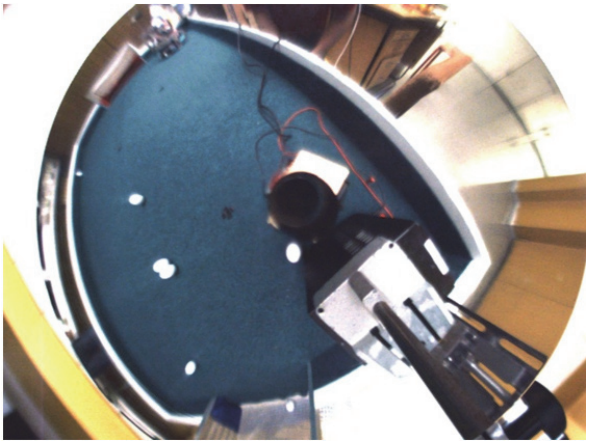
Robot näeb maailma läbi hüperbolidpinnalise peegli, mis asetseb kaamera kohal. Joonisel 2 on toodud vaade väljakule roboti pea kohalt. Robot on väljaku keskel olev hall valge keskmega ring. Joonisel 3 on toodud renderdus disainist: nagu näha, moonutab kaamera kohal paiknev peegel ruumi. Jooniselt 4 leiab läbi roboti kaamera saadava töötlemata pildi. Selliselt pildilt objekte tuvastades peab arvestama, et objektide kuju sõltub oluliselt objektide paigutusest roboti suhtes. Joonisel 4 paiknevad valged laigud on pallid. Peale kaamera kalibreerimist peab

olema võimalik määrata pildilt valitud pikslite vahekaugust reaalses maailmas eeldades, et pikslid asuvad samal tasandil.

Joonis 2 Simuleeritud olukord Robotex-i robootikavõistluse mänguväljakust. Pildi keskel asuv hall ring on võistlusrobot. Selle sees asuv heledam ring on hüperboloidpinnaga peegel. Rohekas ala on väljak ja valged ringid esindavad palle.



Joonis 3 Robotex-i 2010 aasta võistlusväljak läbi roboti silmade. Tegu on renderdusega. Tumedast alast välja poole jääv osa on peegli kinnitus. Pildi keskel on roboti enda peegeldus ning valged ümarad objektid on golfipallid.



Joonis 4 Pildil on Robotex-i võistlusväljak. Must ring pildi keskel on objektiivi peegeldus. Valged ümarad objektid on golfipallid. Vasakul üleval on näha võistlusväravat. Rohekassinine ala on vaibaga kaetud võistlusväljak.

1.2 Ülesande mudel

Töös tegeletakse olukorraga, kus sisendpildiks on 640x480 lahutusega pilt, millel on näha kalibratsioonimuster. Sisendpilt genereeritakse POVRay tarkvara poolt ette antud maailma mudelist. Kalibratsioonimustrit üritatakse tuvastada kasutades pildilt nurkade otsimise meetodit. Mustriks on valgest paberist ruut, mis asub tasapinnaliselt põrandal. Tavapärase maleruudustiku asemel kasutame valget ruutu tema lihtsa tuvastatavuse tõttu. Selles töös on rõhk mitmelt järjestikuselt kaadrilt saadud info sidumisel. Meetodi täpsem kirjeldus on esitatud töö järgmistes peatükkides. Peale edukat kalibratsioonimustri tuvastust soovitud arvult kaadritelt koondatakse kõik saadud tingimused ja leitakse vastav teisendus pildi ja põranda vahel. Selle teisenduse abil saame teada soovitud kaugused, kui kasutame lisateadmist kalibratsiooniruudu küljepikkuse kohta. Joonis 5 esitab kalibreerimisprotsessiks vajaliku andmete liikumise järgnevuse.

Joonis 5 Üldine katseskeem.

2 Optilised lahendused

Robootika seisukohalt on kõigepealt vaja leida sobiv viis ümbritseva keskkonna nägemiseks. Eesmärk on näha võimalikult suurt osa mängust ja seega on vaja võimalikult laia vaatenurgaga süsteemi. Lisaks tuleb arvestada reaalajalisuse nõuetega. Järgnevalt vaatame erinevaid võimalusi kaamera ja optika valikuks.

2.1 Liikuv kaamera

Esiteks saab kasutada ühte kitsama vaatenurgaga kaamerat ja seda liigutada, saavutades nii üle pikema aja keskmistades laiema vaatenurga. Joonis 6 esitab vastava kaamera näidise. Sellise süsteemi puudus on riistvara keerukus ja potentsiaalselt ka kaamera liikumisest tekkiv moonutus. Nende puuduste väljendumine sõltub tugevalt süsteemi ajakriitilisusest.

Joonis 6 Mootoriga suunatav kaamera [1].

2.2 Lainurkobjektiiv

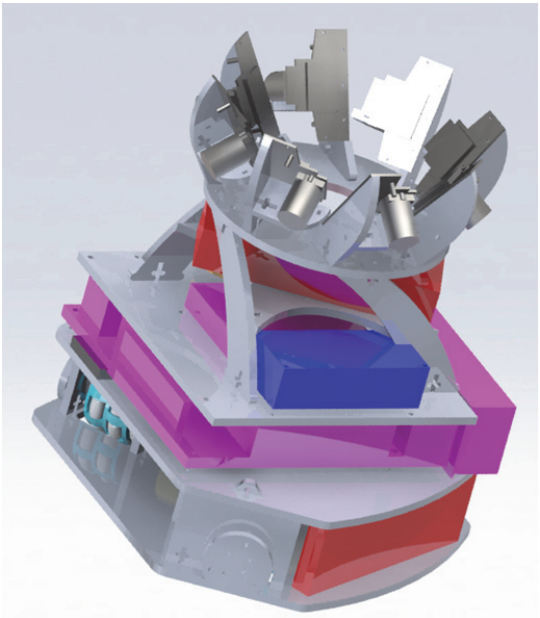
Teine võimalus on kasutada väga laia vaatenurgaga objektiive nagu Tamron-i objektiiv, mis on toodud joonisel 7. Sellise süsteemi miinuseks on väga suur punktitihedus mingis pildi osas ja sealne informatsioon võib jääda kasutamata. Selline omadus võib loomulikult teatud süsteemides vajalik olla. Lisaks sellele on ainult kalasilma objektiiviga võimalik saavutada võrreldavaid vaatenurki kaamera-peegel süsteemiga.

Joonis 7 Fikseeritud fookuskaugusega (2,2 mm) lainurkobjektiiv [2].

2.3 Mitu kaamerat

Kolmas võimalus on mitme eri suunda vaatava kaamera kasutamine. Sellise süsteemi puuduseks on vajadus kasutada mitut kaamerat ja rohkem ühendusi kesksesse süsteemi (nt.

arvutisse). Süsteem nõuab üldiselt ka rohkem ruumi, kui ühe kaamera kasutamine. Näitena on joonisel 8 toodud renderdus sellist süsteemis kasutavast robotist.

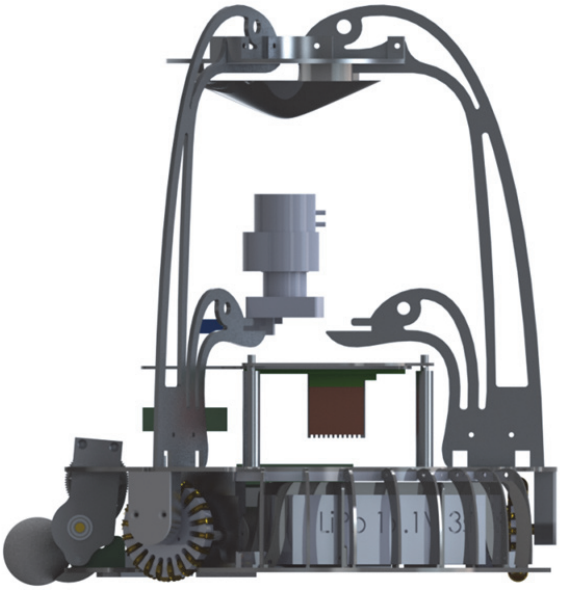


Joonis 8 Süsteem, milles kasutatakse ümbruskonna tajumiseks seitset kaamerat (mustad objektid ülemises servas). Kasutatud autori (Kalle-Gustav Kruus) loal.

2.4 Kaamera ja peegel

Neljandana on võimalik vaatenurga parandamiseks kasutada objektiivi ja peegli süsteemi. Analoogset peeglit kasutatakse liikluses halva nähtavusega tänavanurkadel ja poodides varaste avastamiseks, näide on esitatud joonisel 9. Näide sellist süsteemi kasutavast robotist on toodud joonisel 10.

Joonis 9 Liiklusohutuse tagamiseks mõeldud kumerpeegel tänavanurgal [3].



Joonis 10 Näide robotist, mis kasutab kaamera-peegel süsteemi. Pildi keskel paiknev hall objekt on kaamera, mis vaatab otse üles hüperboloidpinnaga peeglile. Mõned kinnitusdetailid on selguse huvides eemaldatud.

Üldpõhimõte on sama - võimaldatakse näha tavalisest oluliselt suuremat ala. Robootikas kasutatakse erinevatest võimalustest tihti seadistust, kus kaamera vaatab alt üles peegelpinnale, mille lõige on hüper- või parabool. Kasutatakse ka keerulisema lõikega peegleid.

Sellise süsteemi eelis on kompaktsus, lai vaatenurk ja küllaltki vaba peegelpinna kuju valik. Robootika seisukohalt on sellise süsteemi miinuseks suur vabadusastmete arv kaamera ja peegli joondamisel. Lisaks on peegli tootmine kallis tooriku töötlemisel nõutava suure täpsuse tõttu.

Üldeesmärk on välja töötada kalibreerimismeetod, mis toimiks halvasti joondatud, mittekorrektse, lokaalsete lohkudega, hüpeboloidpinnaga peegliga. Käesolevas töös selleni ei jõuta, aga üritatakse eskiisi tasemel ära katta kogu lahendus.

3 Olemasolevad tarkvaralised lahendused

Kaamerate kalibreerimise ülesannet on lahendatud väga paljudel eri viisidel mitmetes teadusasutustes. Kaameraid on paljude erinevate disainidega ning mitmesuguste kasutusvaldkondade ja prioriteetidega.

3.1 Olemasoleva tarkvara iseloomustus

Erinevad lahendused saab jagada 3 peamiseks kasutusgrupiks süsteeminõuete järgi: C/C++ põhinevad üldised pildi- ja videotöötlusteegid (OpenCV), arvutuskeskkonna Matlab-i lisateegid (OcamCalib) ning iseseisvad programmid (DLR Camera Calibration Toolbox). Teiseks oluliseks erinevuseks erinevate lahenduste vahel on täpsuse ja kalibreerimise keerukuse vahekord. Selle järgi jaotuvad lahendused kahte suurde gruppi. Ühed suudavad tuvastada kalibratsioonimustri automaatselt ja seejärel sooritada ka kalibratsiooni. Teised oskavad ainult ühte kahest. Üldiste pildi- ja videotöötlusteekide puhul olenevad võimalused loomulikult ka kasutaja otsustest programmi kirjutamisel.

Enamiku viidatud tööriistade täpsus sõltub oluliselt moonutuse sümmeetrilisusest ning seetõttu ka kaamera ja peegli joondamistäpsusest. Optimeeritakse pigem ühest või paarist kaadrist info kätte saamisele, mitte aga asjaolule, et videopildist on võimalik kerge vaevaga saada oluliselt rohkem kaadreid. Samuti on lahendatud üldisemat ülesannet, kui käesolevas töös on kirjeldatud. Kalibratsioonimustri tuvastamisel leitakse moonutuse järgi mustri asukoht ruumis [4].

Eesmärk on saavutada võimalikult suur sõltumatus peegli tootmisel ja paigutusel tekkivatest ebatäpsustest. See lihtsustaks oluliselt ka peegli tootmise protsessi – peegelpinna geomeetria ei peaks nii täpselt paigas olema. Sõltumatuse saavutamiseks tuleks kaamera süsteemi kalibreerida nii, et iga piksli kohta tekiks eraldi kirje. Olemasolevad tööriistad üritavad aga enamasti tuletada süsteemi geomeetrilisi parameetreid (*camera resectioning)* ja lähtuvad tihti eeldusest, et moonutus on isotroopne.

Lõppkokkuvõttes usub töö autor, et Robotex-i ülesande lahendamiseks on olemasolevatest vahenditest parimad: OpenCV, OcamCalib ja Camera Calibration Toolbox for Matlab. Neid kasutatakse väga aktiivselt kaamerate kalibreerimiseks ja seetõttu on tekkivaid probleeme lihtsam lahendada. OpenCV on toodud esimese valikuna seepärast, et tema kasutamiseks ei ole vaja tasulise programmi olemasolu ja tal on liidesed mitmesse laialt kasutatavasse programmeerimiskeelde. Ülevaatlikkuse huvides kirjeldame rohkem lahendusi.

3.2 Üldised pildi- ja videotöötlusteegid

OpenCV. OpenCV on universaalne teek pildi- ja videotöötluseks, mis sisaldab endas üle 2500 algoritmi [5]. Tegu on väga aktiivselt arendatava teegiga, mille põhiline kasutusvaldkond on reaalajalised rakendused. OpenCV on ka ROS-i (*Robot Operating System*) operatsioonisüsteemis kasutatav baasteek. Põhiliselt arendatakse kasutades C/C++. Lisaks sellele on olemas liidesed keeltele Python ja Java. Siia jõuavad üldiselt algoritmid, mis on ennast juba praktikas tõestanud.

VXL. VXL on C++ tehisnägemise teekide kogumik [6]. Sisaldab algoritme nii pildi- kui ka videotöötluseks. Kogumikku haldab ja arendab grupp teadlasi erinevate ülikoolide teadusgruppidest. On oluliselt vähem tuntud kui OpenCV.

OpenTL. OpenTL on suunatud ilma markeriteta objektide tuvastamisele [7]. Arendatud kasutades C++. See teek on oluliselt suunatuma kasutusvaldkonnaga kui OpenCV või VXL. Funktsioone saab kasutada paralleelarvutustes. Saab kasutada ka kaamerate kalibreerimiseks.

3.3 Arvutuskeskkonna Matlab lisateegid

Camera Calibration Toolbox for Matlab. Üks tuntum kaamera kalibreerimise vahend. Sellel või selle osadel põhinevad mitmed teistes ülikoolides arendatud tööriistad [8] [9][10][11][12]. Töötab keskkonna Matlab versioonidega 5.x – 7.x[4]. Lisaks on väikeste modifikatsioonidega võimalik kasutada ka Octave keskkonnas [13]. Viimane omadus on loomulikult põhiliselt tingitud Octave disainist. Arendatud California Tehnikaülikoolis (edaspidi *CalTech*).

OcamCalib. Väga laialdast kasutust leidnud tarkvara. Osaliselt inspireeritud *CalTech*-is arendatud Matlab-i moodulist [14]. Suudab tuvastada radiaalse moonutuse keskpunkti ka juhul, kui peegli servasid ei ole näha. On seetõttu käepärane vahend kaamera ja peegli joonduse esmasel hindamisel [8].

Omnidirectional Calibration Toolbox. Järjekordne tööriist, mis on oma alguse saanud *CalTech*-is arendatud Matlab-i laiendusest [10]. Kalibreerimiseks on vaja maleruudustikku, mille otsimine käib poolautomaatselt. Igal kaadril tuleb märkida kalibratsioonimustri neli välimist nurka, ülejäänud nurgad tuvastab programm iseseisvalt.

EasyCamCalib Software. Põhiline kasutusvaldkond on endoskoopide kalibreerimine. Olenemata oma põhikasutusvaldkonnast sobib programm ka robootikas kasutamiseks ja on hea näide robootika seostest teiste valdkondadega. Endoskoobile paigaldatakse objektiiv vahetult enne protseduuri ja seetõttu peab kalibreerimisprotsess olema lihtne, kiire ning alles

seejärel täpne. Just seepärast on mindud kalibreerimisel ühe pildi kasutamise teed. Eeldatakse mustri tasapinnalisust. Täpsus on võrreldav mitut pilti kasutavate algoritmidega [15] [16]. Kalibratsioonimustri nurkade tuvastus töötab automaatselt.

Camera calibration toolbox for Matlab. Selle laienduse arendamisel on rõhutudeeldustele, et pildil esinev müra ei ole alati ühtlase juhusliku jaotusega ning moonutuse mittehomogeensusele kogu ruumi suhtes [17] [18]. Ei võimalda kalibratsioonimustri automaatset tuvastamist. Siin töös esitatud kaamera mudelit on kasutatud ka teistes programmides. Viimane uuendus tuli aastal 2000.

GML MatLab Camera Calibration Toolbox. Ka see tööriist põhineb *CalTech*-is arendatud moodulil [11]. Tegu on *CalTech*-i teegi mingi vanema versioonilaiendusega, mis võimaldab automaatset kalibratsiooniobjektide tuvastust. Toetatud on tarkvara Matlab versioonid 6.5 ja 7.0 ning operatsioonisüsteemina on testitud erinevaid Windows-i versioone.

Multi-Camera Self-Calibration. Tööriist, mis on loodud mitme kaameraga süsteemide kalibreerimiseks [19] [20]. Süsteemis peab olema vähemalt 3 kaamerat. Kalibreerimismustrit ei ole vaja. Kasutatakse käes hoitavat laserit, mille täppi tarkvara tuvastab. 16 kaameraga saavutati vähem kui 30 minutit väldanud kalibreerimise käigus kaamerate koordinaatsüsteemide vaheline täpsus $\frac{1}{5}$pikslit [20]. Ei ole mõeldud kaamera-peegel süsteemide kalibreerimiseks. Oma ehituselt on tegu süsteemiga, mis kasutab kõige lihtsamat võimalikku kalibratsioonimustrit üle paljude kaadrite. Ühe kaameraga süsteemides ei ole võimalik ilma lisainformatsiooni kasutamata ühe punkti abil kaamerat kalibreerida.

Camera Calibration Toolbox for Generic Lenses. Kalibreerimiseks piisab, kui tuvastada kalibratsiooniruudustik ühest kaadrist, aga täpsuse saavutamiseks on soovitav mitme kaadri kasutamine [21]. Kalibreerimiseks kasutatakse tavapärasest erinevat tasapinnalist mustrit korrapäraste valgete ringidega, mis paiknevad mustal taustal. Mustrid on kodulehelt kättesaadavad. Selle rakenduse juures on eraldi mõeldud juhule, kus kalibratsioonimustrit ei trükita paberile, vaid vaadatakse kaameraga otse monitorilt. Sellist lahendust käesolevas töös toodud ülesande lahendamiseks hästi kasutada ei saa, sest kalibreeritakse põranda suhtes ja see eeldaks väga õhukese ning hea vaatenurgaga monitori kasutamist. Toetatud on Matlab-i versioonid alates 6.5-st. Lisaks sellele on töötamiseks vajalikud Matlab-i tasulised laienduspaketid Image Processing Toolbox ja Optimization Toolbox [22].

MetroVisionLab. Selles tarkvaras koondatakse mitmeid varem välja arendatud mudeleid ja algoritme [23]. Võimaldab võrrelda erinevate lahenduste tulemuslikkust samadel sisendandmetel. Põhirõhk ongi erinevate lahenduste võrdleval analüüsil ning sobib seejuures loomulikult ka kaamera kalibreerimiseks. Vajab töötamiseks vähemalt Matlab 7.0. Viimane uuendus jaanuar 2009.

FAUCCAL. Selle Matlab-i laienduse puhul rõhutakse kalibratsiooniprotsessi automaatsusele [24] [25]. Kalibreerimiseks kasutatakse tavapärast maleruudustikku. Nurkade tuvastamisel kasutatakse Harris-e meetodit [26]. Erineb teistest lahendustest just oma nurgatuvastusprotsessi poolest. Tuvastamise järel leitakse võimalike nurkade hulgast mediaan, põhipunkt ning teiste nurkade verifitseerimine käib selle punkti suhtes [27]. Keskmist ei kasutata tema suurema tundlikkuse tõttu mustrist väljas asuvate valede punktide suhtes. Ülejäänud punktide jaoks arvutatakse kaugus põhipunktist.

EasyCal. Tööriist, mis on loodud mitme kaameraga süsteemide kalibreerimiseks. Kasutatakse kalibratsioonimustreid ja valgusallikat [28]. Eeldatakse vähemalt 1000 kaadrit summaarselt, mis sisaldavad valgusallikat. Soovitatav on 5000 kaadrit. Lisaks sellele on vaja kaadreid maleruudustikust nii, et muster on nähtav vähemalt kahest kaamerast. Selliseid kaadreid peaks olema umbes 40 iga kaamera kohta, kust terve muster näha on. Tegu ei ole puhta Matlab-i põhise rakendusega. Osa arendust on tehtud .NET keskkonnas. Arhitektuurilt on moodul nagu *CalTech*-i arendus kombineerituna Multi-Camera Self-Calibration paketiga.

3.4 Iseseisvad valmisprogrammid

DLR Camera Calibration Toolbox. Tööriist koosneb kahest programmist ja sobib ka stereokaamera kalibreerimiseks [29]. Esimesena DLR CalDe, mida kasutatakse mingi konkreetse kalibratsioonimustri punktide otsimiseks. Sellega on jäetud suurem vabadus kasutada nurkade tuvastamiseks mõnda muud vahendit. Ning teisena DLR CalLab, mida kasutatakse kaamera kalibratsiooniparameetrite määramiseks. Kalibreerimiseks on vaja ka IDL virtuaalmasinat, milles seda programmi jooksutada. Programmi saab kasutada tasuta.

Tclcalib. Tarkvara, mis koondab tervet hulka varem arendatud programme. Erinevate programmide sidumiseks on kasutatud Tcl keelt [30]. Operatsioonisüsteemid Irix ja Microsoft Windows 2000 (Win2k). Keskse kaamera mudelina kasutatakse J. Heikkila mudelit. Tegu on pigem arendaja isiklikuks teadustööks mõeldud tööriistaga ja seetõttu on kasutajaliidese kujundamisel lähtutud rohkem isiklikest eelistustest ja vähem üldistest tavadest. Tarkvara on võimeline kalibratsioonimustri ise leidma. Kalibreerimiseks kasutatakse korrapäraseid musti ringe valgel taustal. Kalibreerimisprotsess on esitatud väga selge järgnevusega.

OMC Camera Calibration Software. Selle programmi puhul on tegu kommertstarkvaraga. Kalibreerimismustrina kasutatakse valget kõvale alusele liimitud paberit, millel asuvad kindla paigutusega mustad ringid ja ringide sees valged täpid, kui lõppkasutuseks piisab tasapinnalisest kalibreerimisest [31] [32]. Keerulisematel juhtudel ehitatakse kalibratsioonimuster lõppkasutusele sarnase mudelina. Mustri printimisel tekitatakse ka mudel

programmi jaoks, mis määrab ära mustri ligikaudse geomeetria [33] [34]. Kalibratsioonimustrit pildistatakse tavaliselt 8 korda nii, et mustri ja sensori vaheline nurk on ligikaudu 45 kraadi. Nendest neljal korral on pildi pikem külg horisontaal- ja neljal vertikaalsihis. Lisaks kaamera asetusele on kirjeldatud ka sügavusteravuse ja ava soovitavat asetust ja selle saavutamist. Sellise tarkvara eripära on tasulisele tarkvarale kohaselt garanteeritud kasutajatugi. Robotex-i haridusliku olemuse tõttu ei ole sellise tarkvara soetamine eriti põhjendatud.

Camera Calibration tools. Operatsioonisüsteemile Windows XP mõeldud tarkvara, mis kasutab sisemiselt suures osas *CalTech*-i matlab-i laienduse tööpõhimõtet [9]. Kõige suurem uuendus on sõltumatus Matlab-ist. Rõhutakse kergesti mõistetava eraldiseisva rakenduse arendamisele. Eeldab DirectX viimast versiooni, aga võib töötada ka variantidega 9.0b-st ja 9.0c-st. Arendus on viimased 4 aastat seisnud.

GML C++ Camera Calibration Toolbox. Programm kirjutati OpenCV 2.1 laiendamiseks. Toetab mitme kalibreerimismustri samaaegset kasutamist, vähendades sellega vajaminevate kaadrite hulka [12]. Testimiseks kasutati põhiliselt operatsioonisüsteeme Windows XP, Windows Vista ja Windows 7. Vajalik .Net Framework 3.5 SP1 ühilduvus.

Microsoft Easy Camera Calibration Tool. Programm on mõeldud radiaalse moonutuse eemaldamiseks [35] [36] [37]. Hinnatakse kaamera parameetreid, näiteks fookuskaugust. Selline parameetrite hindamine on võimalik ka enamuses teistes programmides. Kalibreerimiseks kasutatakse tasapinnalist kalibreerimismustrit. Kalibreerimiseks on vaja vähemalt kahte erinevat kaadrit kalibratsioonimustrist. Kalibreerimismuster leitakse automaatselt. Viimane uuendus oli aastal 2001.

Tele2. Tarkvara võimaldab nii kaamera kalibreerimist kui ka 3D objektide rekonstruktsiooni [38] [39]. Kaamera kalibreerimiseks kasutatakse mustrit, mis on kokku pandud kolmest täppidega paberist nii, et iga kaks mustrit on teineteise suhtes risti. Nurkade tuvastus on poolautomaatne. Kasutaja peab määrama tuvastusalgoritmi parameetreid kuni soovitud tulemuse saavutamiseni. Sellise kalibreerimismustri valmistamise vajadus teeb kalibreerimisprotsessi oluliselt ebamugavamaks, kui lihtsalt ühe kõval alusel mustri kasutamine.

4 Testandmed

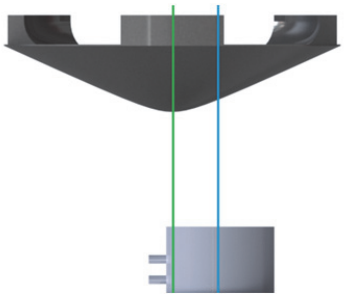
Programmi arendamise lihtsustamiseks genereeriti testandmed kasutades tarkvarapaketti POVRay 3.6. See tarkvara kasutab renderdades *raytracing* (pildi pikslitest lähtuvate kiirte meetodi) põhimõtet. Genereeritud andmete kasutamine võimaldab programmi käitumist

paremini analüüsida ja alustada lihtsama ülesande lahendamisest. Alternatiivse võimalusena võiks kohe alguses kasutada sisendina robotilt saadud pilti.

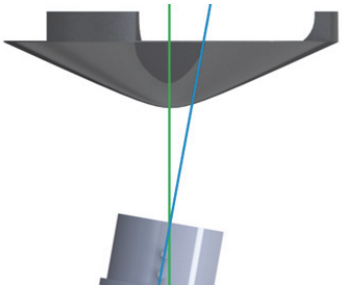
Selline lähenemine tasus kohe antud töös nurgatuvastusmeetodit kirjutades ära. Esialgu sai kirjutada meetodi, mis töötas ainult kergelt moonutatud mustri korral ja seejärel sai sisendit keerulisemaks muuta ja funktsioone vastavalt sellele täiustada. Seega sai tarkvara arendada lähtudes ühest *test-driven development*-i põhimõtetest. Robotilt tuleva video kasutamine oleks eeldanud koheselt kõigi moonutustega arvestamist.

4.1 Kaamera-peegel süsteemis tekkivad moonutused

Moonutusi, millega tuleb arvestada, on palju ja kalibreerimise käigus võivad olukorda halvendada kaamera värvide suhtes kalibreerituse probleemid. Peegli geomeetria – siin all mõeldakse peegli soovitud kuju. Peegelpinna ebaühtlus – valguse hajumise ellimineerimiseks oleks vajalik pinna karedus alla $\frac{1}{4}$ valguse lainepikkuse. Rahuldava teravusega pilt on võimalik saavutada oluliselt suurema pinnakareduse juures. Peegli ja objektiivi telje nihe on illustreeritud joonisega 11. Joonis 12 esitab peegli ja objektiivi vahelise kaldumise juhu. Ebaühtlane valgustatus raskendab nurkade tuvastamist ja suurendab valepositiivsete nurkade hulka. Omaette moonutuste rühma moodustavad objektiivi ehitusest sõltuvad häired: objektiivi geomeetria, materjali mittehomogeensus, joondus kaamera sensori suhtes, fookuskaugus, sügavusteravus, neeldumisnäitaja ning ava suurus. Moonutusi põhjustavad veel kaamera-peegel süsteemi liikumine ruumiobjektide suhtes ja sensorite tööpõhimõttest tulenevad eripärad olenevalt, kas kasutatakse CCD (laengsidestusseadis) või CMOS (komplementaarne metall-oksiid-pooljuht) tüüpi kaameraid. Erinevate moonutuste kohta on täpsemalt kirjutatud A. Hornbergi tehisnägemise õpikus [40].



Joonis 11 Peegli ja objektiivi telje nihe võimendatud kujul. Roheline joon kulgeb mööda peegelpinna sümmeetriatelge ja sinine joon kulgeb mööda objektiivi läätsede sümmeetriatelge.



Joonis 12 Peegli ja objektiivi telje kaldumine võimendatud kujul. Roheline joon kulgeb mööda peegelpinna sümmeetriatelge ja sinine joon kulgeb mööda objektiivi läätsede sümmeetriatelge.

4.2 Renderdaja valik

POVRay 3.6 on tööks valitud laia leviku ja vaba kättesaadavuse tõttu. Lisaks sellele on POVRay taga aktiivne grupp arendajaid, mis annab kindlustunde tarkvara vigade eemaldamise suhtes. POVRay-st on väljas ka versioon 3.7, mis toetab mitme protsessori kasutamist, aga see versioon on alles beeta väljalase. Lisaks sellele puudus vajadus mitme tuuma kasutamise järele.

POVRay-l on veel üks väga oluline omadus – identse testpildi genereerimiseks piisab konfiguratsioonifailidest. Nii on võimalik teistel tarkvaraarendajatel kasutada täpselt samu testandmeid. See oleks võimalik ka testpiltide säilitamisel, aga selle meetodid miinuseks on asjaolu, et piltide pakkimisel on valdavalt kasutusel kadudega pakkimise algoritmid – programmiteksti puhul on selliste pakkimisalgoritmide kasutamine oluliselt haruldasem ja vähem taotluslik.

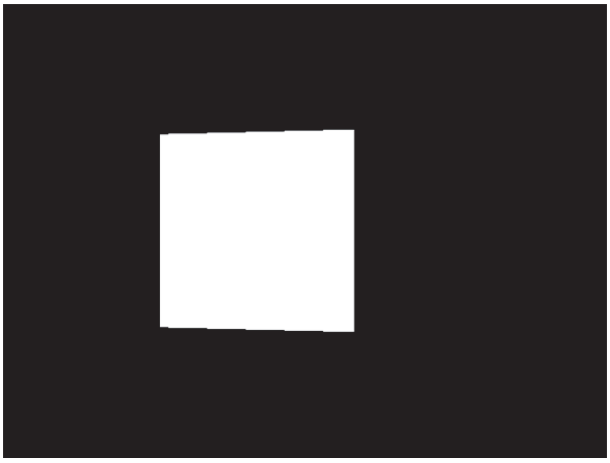
Tarkvara võimaldab ka liikumise simuleerimist. Kaadrisagedus määratakse kestvuse ja kaadrite arvu kaudu. Väljundiks on siiski iga kaader eraldi. Selle põhjenduseks on asjaolu, et kaadritest video kokku panemiseks on piisavalt muid vahendeid ja nii saab keskenduda renderdusvahendite arendamisele. POVRay väljundist saab teha video kasutades programmi mencoder.

4.3 POVRay tööpõhimõte

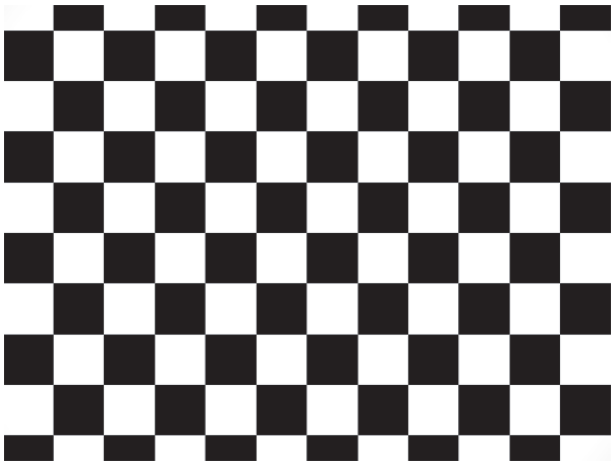
Programm vajab sisendina .ini ja .pov lõpulisi faile. Töös kasutatud failide näited on toodud lisades 1 ja 2. Iga testpiltide komplekti genereerimiseks seadistati tarkvara natuke erinevalt. Esimeses failis on programmi globaalsed seadistused. Seal kirjeldatakse näiteks väljundfailide nimed, kaadri suurus ja arv. Täpsemalt vaata lisast 1. Teises failis kirjeldatakse ruum, milles arvutusi tegema hakatakse. Ruumi kirjelduse alla käivad kõik objektid koos oma tekstuuriga. Lisaks sellele kirjeldatakse kõik valgusallikad ja kaamerad koos oma täpse asukohaga. Programmi käivitamisel hakatakse *raytracing* põhimõttel kaadreid genereerima.

4.4 Genereeritud andmed

Renderdatud objektid on üldiselt lihtsad ja keskmise lauaarvuti ühel tuumal kulub ainult perspektiivi arvestamisel iga 640x480 piksli suuruse kaadri loomiseks umbes 1 sekund. Lahutuse valimisel on arvestatud nii reaalse riistvara kui ka testimiseks kuluva ajaga. Tarkvara täpsemal häälestamisel saab lahutuse POVRay konfiguratsioonifailis lihtsalt ära muuta ja testandmed uuesti genereerida. Testpildi näiteks sobivad hästi joonised 13 ja 14. Joonisel 14 on toodud pilditöötluses väga tihti kasutatav kalibratsioonimuster. Kui simuleerida ka kumerpinnalt peegeldumist ja rakendada *spatial anti-aliasing* pildi silumise meetodit, siis kasvab ühele kaadrile kuluv aeg kiiresti üle 10 sekundi. Sellise pildi näideteks sobib joonis 18.



Joonis 13 Kalibratsioonimuster vaadatuna läbi *pinhole* tüüpi kaamera. Kalibratsiooniruudu servad on sakilised, sest pildile ei ole rakendatud *spatial anti-aliasing* (vektorgraafikast rastergraafikasse teisendamisel tekkiva efekti silumise) meetodit.



Joonis 14 Malelaua stiilis kalibratsioonimuster. Selliste mustrite väljatrükke kasutab kalibreerimiseks enamus siin töös vaadeldud alternatiivsetest programmidest.

4.5 Edasine uurimistöö

Siin esitatu on vajalik ülesande täielikuks lahendamiseks, aga see ei kuulu käesoleva töö skoopi. Esimese ülesandena tuleks teha mudelid hüperboloidse peegli jaoks. Mudelid tuleks luua nii täpsest kui ka moonutatud peeglist. Selleks otstarbeks on ilmselt kõige mõistlikum kasutada SolidWorks tarkvara, milles ka kogu ülejäänud robot disainiti. SolidWorksis loodud mudeli saaks viia POVRay-le sobivasse formaati kasutades tarkvara PoseRay.

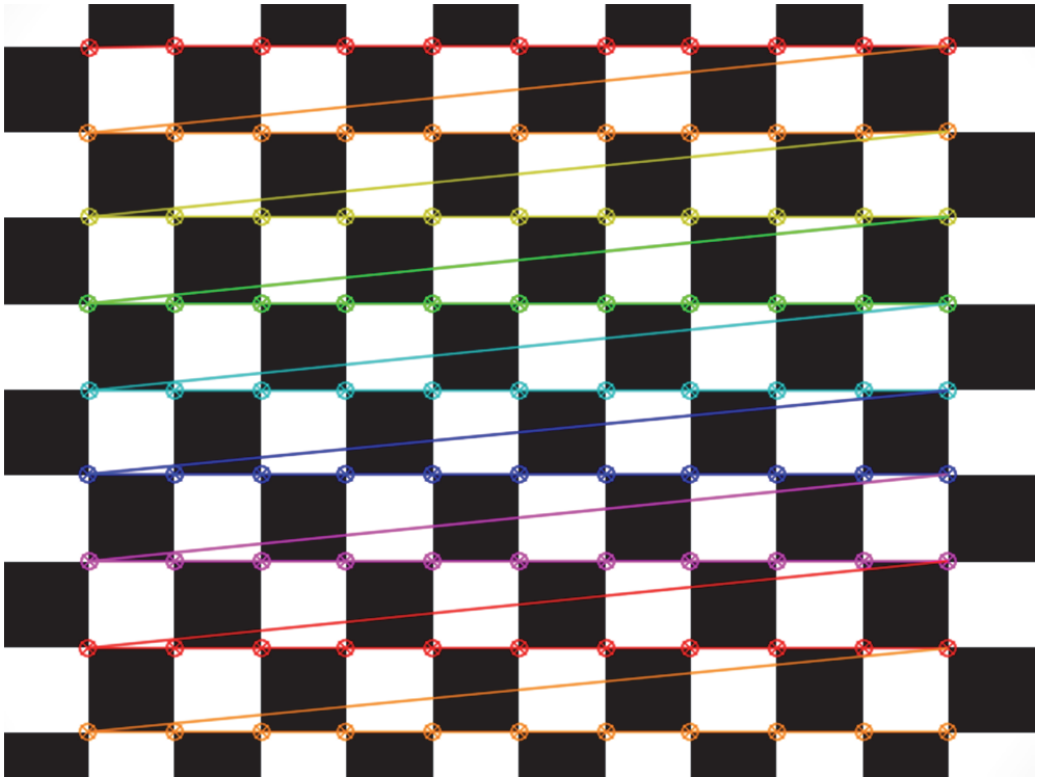
Lisaks sellele tuleks luua mudelid olukordadest, kus simuleeritakse erinevaid optikaga seotud probleeme. Näiteks oleks vaja luua mudelid olukordadest, kus on toimunud kaamera ja peegli telgede nihe, telgede kalle ning olukordadest, kus esinevad mõlemat tüüpi probleemid.

Loodud mudelitest tuleks renderdada suur kogus testpilte. Lisaks loodava tarkvara testimisele annaks see võimaluse võrrelda enda loodud vahendit erinevate teiste loodud kalibreerimismeetoditega. Tänu testandmete simuleerimisele annaks see võimaluse teha järeldusi erinevate mudelite omadustest.

5 Nurgatuvastus

5.1 OpenCV malelaua otsimise algoritm

Töö varajases staadiumis katsetati ka OpenCV sisseehitatud malelaua tüüpi kalibratsiooniruudustiku tuvastamise vahendit, näide tulemusest on joonisel 15. Selle meetodi eelis on täpne ja kiire nurkade tuvastamine. Kiirus ei ole käesolevas töös käsitletava ülesande juures kõige olulisem, sest kalibratsiooni saab enne võistlust ära teha. Miinuseks on see, et funktsioonile tuleb täpselt ette anda, mitu sisemist nurka kalibratsioonimustril on – antud juhul 9x11. Keerulisemates olukordades võib kalibratsioonimuster tuvastamata jääda, sest näiteks kõige kaugemal asuvat nurka ei suudeta tuvastada. OpenCV meetod on ka tundlikum kalibratsioonimustri ebaühtlase valgustatuse suhtes [41]. Lisaks sellele annab nurgatuvastaja ise kirjutamine sügavamaid teadmisi sellist tüüpi ülesannete lahendamisest.



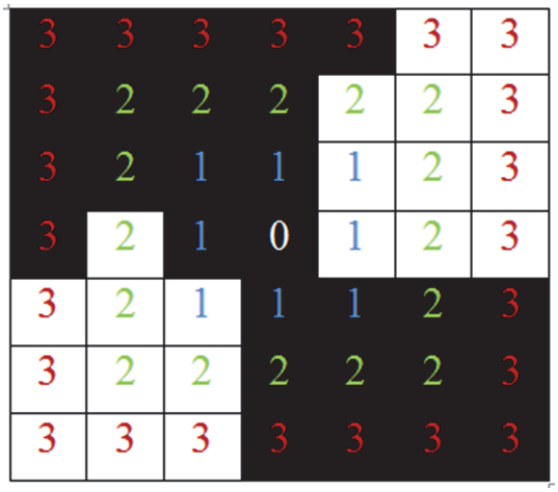
Joonis 15 OpenCV FindChessboardCorners() meetodi poolt tuvastatud kalibratsioonimuster.

5.2 Kohandatud kihilise nurgatuvastuse meetod

Nurkade tuvastamiseks vajaliku programmi kirjutamiseks kasutatakse lähteallikana algoritmi, mida on kirjeldatud artiklis [41]. See meetod on oma ehituselt väga töökindel viis nurkade leidmiseks [42]. Kombineerituna samas artiklis [41] kirjeldatud ruudustiku leidmise algoritmiga võimaldab välja valida ainult need nurgad, mis kuuluvad kalibratsiooniruudustikku. Meetod on välja töötatud projektor-kaamera süsteemide jaoks.

Selliseid süsteeme kasutatakse olukorras, kus projektsiooni pind ei ole tasapinnaline ja projektori optiline peatelg ei ühti projektsioonipinna normaaliga. Lihtsamate nihete korral peatelje suhtes on tavaliselt võimalik kasutada projektorisse sisse ehitatud vahendeid. Keerulisematel juhtudel - kaasa arvatud kõik projektsioonipinna ebaühtlused - tuleb kasutada teistsuguseid meetodeid. Lahendatav ülesanne on oma olemuselt väga sarnane kaamera-peegel süsteemi kalibreerimisega mingi kindla tasapinna suhtes. Mõlemal juhul on tarvis teada, kuidas tuleb pilti moonutada, et lõpptulemus oleks võimalikult ühtlane.

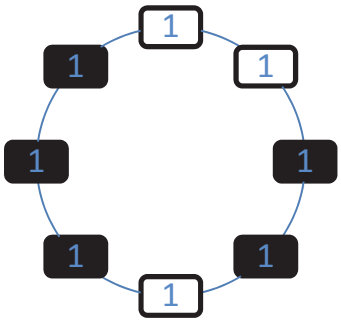
Nurkade otsimisel eeldatakse, et sisendpildil leiduvad otsitavad nurgad. Sisendpilt on must-valge, mis tähendab, et oluline on ainult intensiivsus. Otsustamaks, kas kontrollitav piksel on nurk, vaadatakse tema naaberpiksleid. Selleks rakendatakse nelinurkset akent, sest ringikujulise akna rakendamine ei anna deformatsioonide tuvastamisel isotroopsuse mõttes eelist [41]. Nagu näha jooniselt 16 jaotub piksli ümbrus kihtideks.



Joonis 16 Nurga suurendamisel näeme analoogset pilti, kus osa pikslitest on mustad ja osa valged. Piksel tähisega null on parasjagu kontrollimisel. Selleks jagatakse teda ümbritsevad pikslid kihtidesse.

Nurga tuvastamisel kombineeritakse kõigist kihtidest saadud tulemus otsuseks, kas tegu on nurgaga etteantud piiride mõttes. Kombineerimine toimub hääletamise põhimõttel: kui piisavalt palju kihte sisaldavad 4 regiooni, siis loetakse piksel nurgaks.

Iga kihti vaadeldakse ühemõõtmelise massiivina. Arvestades kalibreerimismustri geomeetriat otsitakse kihist 4 regiooni. Kiht võib alata ja lõppeda sama regiooniga, sest nurk võib olla mistahes rotatsiooniga. Seetõttu rakendatakse regioonide tuvastamisel ja müra eemaldamisel ringpuhvriga sarnast põhimõtet, pikslite esitus on toodud joonisel 17.



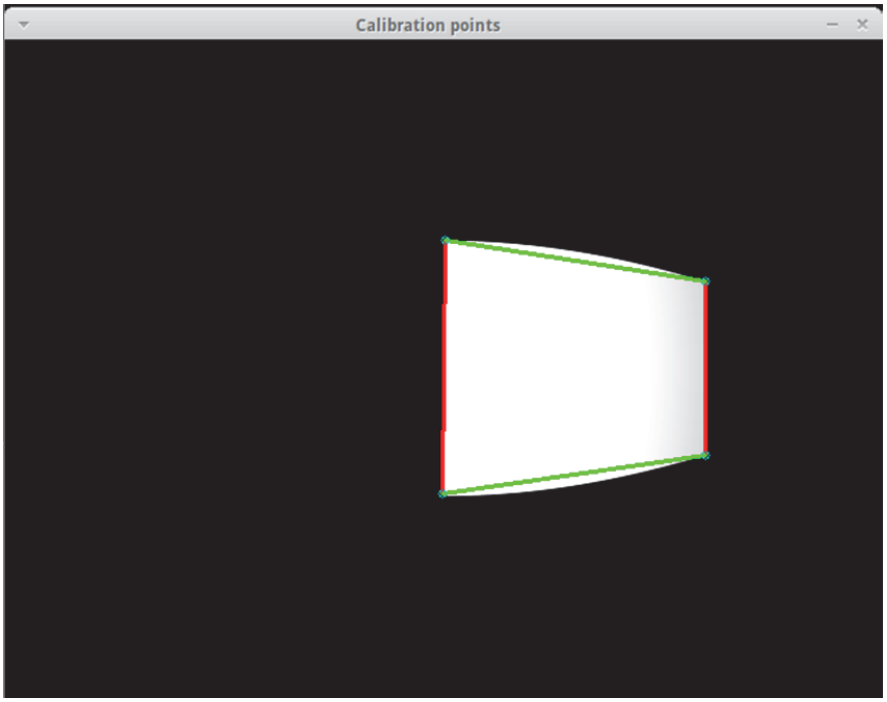
Joonis 17 Keskmist pikslit ümbritsev esimene kiht ringpuhvrina.

Eeldusel, et tegu on maleruudustikuga, peaksmoonutamata pildi iga nurga keset (keskmine piksel) ümbritsev regioon katma pidevalt ligikaudu $\frac{1}{4}$ kihti. Arvestades, et pildil esinevad antud tööülesande püstituse korral moonutused, lubame regioonid, mis katavad vähemalt $\frac{1}{10}$ kogu kihist.

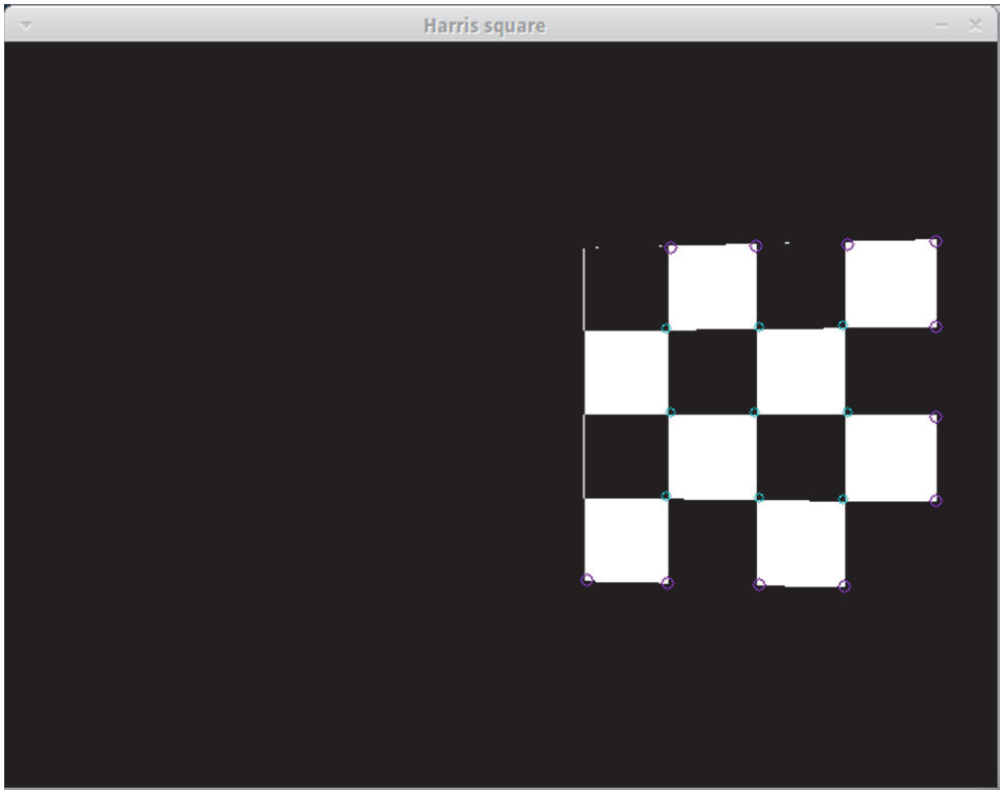
Artiklis rakendatakse järgmisena teadmist, et nende nurkade hulgas peaks olema alamhulk nurki, mis kuuluvad kalibratsioonimustrile. Käesolevas töös seda osa ei käsitleta, sest pannakse rõhku mitmest kaadrist saadud infole. Samas jääb võimalus kasutada tulevikus kalibratsioonimustrina maleruudustikku, mis võimaldaks rohkema info saamist ühest kaadrist.

5.3 Tulemused

Implementeeriti lihtsustatud versioon kihilisest nurkade tuvastamise algoritmist ja rakendati seda genereeritud kalibratsioonimustrile. Programm on Internetis vabalt ligipääsetav link on toodud lisas 3. Tuvastati edukalt nii üksik ruut, tulemus joonisel 18, kui ka maleruudustik, tulemus joonisel 19. Katsetati ka OpenCV sisseehitatud Harris-e nurgatuvastusmeetodit; see tulemus on näha joonisel 19. Ka viimasega oleks võimalik peale pikemat seadistamist tuvastada kõik antud testpildil toodu nurgad, aga sellise meetodid kasutamine raskendaks tulevikus programmi täpset seadistamist. Programmi tekst on esitatud lisas nr. 3.



Joonis 18 Kihilist nurgatuvastusmeetodit kasutades tuvastatud ruut. Ruudu küljed on paari kaupa värvitud. Kalibratsiooniruut on moonutatud, sest seda vaadeldakse kujutisena kerapeeglil.



Joonis 19 Lillade ringidega on tähistatud OpenCV Harris-e detektorit kasutades saadud tulemus. Sinakasroheliste ringidega märgitud nurgad tuvastas käesoleva töö tarbeks kirjutatud programm.

6 Pikslite sidumine

Järgnevalt esitatakse üks võimalik lahendus kaadritelt tulnud info tõlgendamiseks ning kaamera kalibreerimiseks. See meetod põhineb bilineaarsel interpoleerimisel. Bilineaarset teisendust kasutatakse pilditöötluses näiteks pildi suurendamise juures vahepealsete pikslite värvide arvutamiseks. On ka oluliselt keerulisemaid algoritme, mis arvestavad rohkemate ümbritsevate pikslite väärtustega.

Bilineaarne seos ei ole tegelikult lineaarne, vaid kahe lineaarse seose korrutis:

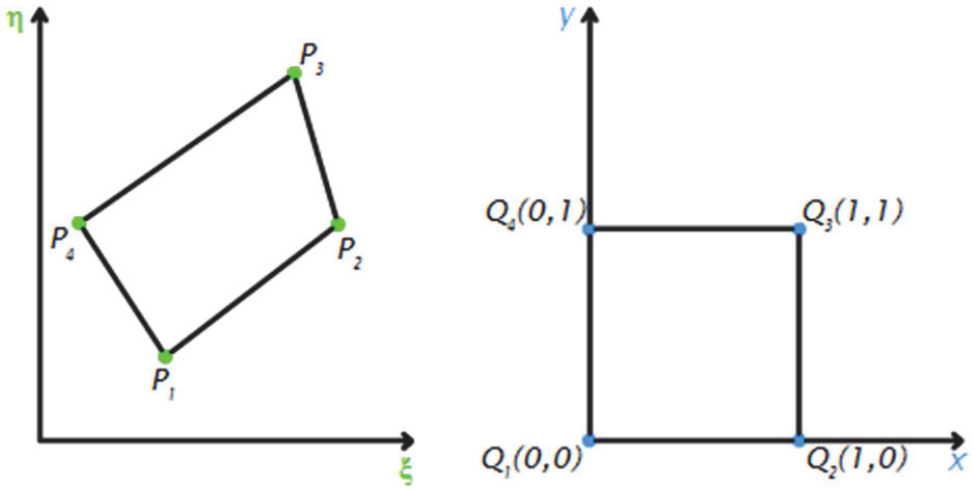
$$F(x,y) = (u_0x + u_1)(u_2y + u_3) = u_0u_2xy + u_0u_3x + u_1u_2y + u_1u_3.$$

Mugavamaks kasutamiseks tähistame kordajad ühe tähega ja nii saamegi bilineaarse seose üldkujul

$$F(x,y) = v_3xy + v_2x + v_1y + v_0.$$

Töös püstitatud ülesandest lähtuvalt on meil teada kujundite asukohad originaalteljestikus. Kaamerast näeme moonutatud pilti, mida mõõdame ξ, η koordinaadistikus. Eesmärgiks on leida bilineaarsed teisendused $\xi = f(x,y)$ ja $\eta = g(x,y)$, mis aitaks originaalpildi vaatluseks teisendada. Joonisel 20 on esitatud telgede tähistused, mida kasutame kogu peatükki vältel.

Originaalteljed: x, y
Vaatluse teljed: ξ, η



Joonis 20 Vasakul pool on toodud vaatluse teljed (kaamera pilt) ja üks kalibratsiooniruut selles teljestikus, punktid P_1 - P_4, ning paremal pool on toodud originaalteljed koos kalibratsiooniruuduga (ühikruut).

6.1 Bilineaarne teisendus originaalist vaatlusesse

Eeldame, et kogu pilt on bilineaarse moonutusega ja ühelt kaadrilt on korrektselt tuvastatud kalibratsiooniruut. Nüüd soovime leida bilineaarsed teisendused, mis väljendavad seost füüsilise maailma ja kaamera pildist saadud observatsiooni vahel. Ehk vaadates joonist 20 soovime leida teisendust, mis viiks punkti Q_i punktiks P_i. Koordinaatide leidmisel eeldame, et: $Q_1 \to P_1;\ Q_2 \to P_2;\ Q_3 \to P_3;\ Q_4 \to P_4$. Sellise eelduse saame teha sest kalibratsiooniruudu tuvastamisel teeme kindlaks punktide paiknemise ruudus.

Eeldusest, et kogu pilt on bilineaarse moonutusega teame, et eksisteerib nii teisendus, mis väljendab punkti ξ-koordinaati kui ka teisendus, mis väljendab punkti η-koordinaati. Esitame need teisendused vastavalt eespool toodud bilineaarse teisenduse üldkujule järgmiselt

$$\begin{cases} \xi = c_3xy + c_2x + c_1y + c_0 \\ \eta = d_3xy + d_2x + d_1y + d_0 \end{cases}.$$

Koondame tundmatud viitamise lihtsustamiseks vektoritesse $\vec{c} = (c_3, c_2, c_1, c_0)$ ja $\vec{d} = (d_3, d_2, d_1, d_0)$. Kaamera kalibreerimiseks kasutame ruudu kujulist paberit. Loeme selle ruudu ühikruuduks ning ning valime teljestiku nii, et üks ruudu nurkadest asub punktis koordinaatidega (0,0) ja teine punktis koordinaatidega (1,1). Paigutus on toodud ka joonisel 20.

Avaldame eeltoodut arvesse võttes observatsioonid $P_1 - P_4$ asendades üldkujus muutujad x_i ja y_i punktile vastava originaali koordinaatidega.
Punkti P_1 koordinaadid on antud eeldustel

$$\begin{cases} \xi_1 = c_0 \\ \eta_1 = d_0 \end{cases}.$$

Punkti P_2 koordinaadid on

$$\begin{cases} \xi_2 = c_2 + c_0 \\ \eta_2 = d_2 + d_0 \end{cases}.$$

Punkti P_3 koorinaadid on

$$\begin{cases} \xi_3 = c_3 + c_2 + c_1 + c_0 \\ \eta_3 = d_3 + d_2 + d_1 + d_0 \end{cases}.$$

Punkti P_4 koordinaadid on

$$\begin{cases} \xi_4 = c_1 + c_0 \\ \eta_4 = d_1 + d_0 \end{cases}.$$

6.1.1 Näidislahendus 1

Olgu originaalteljestikus ruut koordinaatidega $Q_1(0,0)$; $Q_2(1,0)$; $Q_3(1,1)$; $Q_4(0,1)$ ja vaadeldavas teljestikus ruut koordinaatidega $P_1(1,1)$; $P_2(7,3)$; $P_3(6,8)$; $P_4(2,6)$. Leiame vastavalt eeltoodud meetodile bilineaarsed seosed nende punktide vahel.

Nendest seostest saame järgmise võrrandisüsteemi:

$$\begin{cases} c_0 = 1 \\ d_0 = 1 \\ c_2 + c_0 = 7 \\ d_2 + d_0 = 3 \\ c_3 + c_2 + c_1 + c_0 = 6 \\ d_3 + d_2 + d_1 + d_0 = 8 \\ c_1 + c_0 = 2 \\ d_1 + d_0 = 6 \end{cases}.$$

Võrrandisüsteemi lahendamiseks kasutame tarkvarapaketi Maple käsku *solve*, mis on mõeldud lineaarvõrrandisüsteemide lahendamiseks. Võrrandisüsteemi lahendiks saame:

$$\begin{cases} c_0 = 1 \\ c_1 = 1 \\ c_2 = 6 \\ c_3 = -2 \\ d_0 = 1 \\ d_1 = 5 \\ d_2 = 2 \\ d_3 = 0 \end{cases}.$$

Lähtudes eeldusest, et teisendus on kogu pildi ulatuses sama olemegi saanud bilineaarse teisenduse, mis väljendab selle pildi iga originaali punkti observatsiooniteljestikus. Punktile P_i vastav teisendus on üldkujul järgmine:

$$\begin{cases} \xi_i = -2 * x_i y_i + 6 * x_i + 1 * y_i + 1 \\ \eta_i = 0 * x_i y_i + 2 * x_i + 5 * y_i + 1 \end{cases}.$$

Ehk lühemalt :

$$\begin{cases} \xi_i = -2 * x_i y_i + 6 * x_i + 1 * y_i + 1 \\ \eta_i = 2 * x_i + 5 * y_i + 1 \end{cases}.$$

6.1.2 Näidislahendus 2

Esimesest näidislahendusest saime vastava bilineaarse seose tundmatute $\vec{c}, \vec{d}$ ning nende abil ka teistenduste üldkuju. Rakendame nüüd teisendust ühikruudule, mis asub kohe esialgse ühikruudu kohal. Ning mille koordinaadid on $Q_5(0,1)$; $Q_6(1,1)$; $Q_7(1,2)$; $Q_8(0,2)$. Avaldame punktide $P_5 - P_8$ koordinaadid näidislahenduses 1 saadud teisenduse kaudu.
Punkti P_5 koordinaatideks saame:

$$\begin{cases} \xi_5 = (-2) * 0 * 1 \ + 6 * 0 + 1 * 1 + 1 = 2 \\ \eta_5 = 2 * 0 + 5 * 1 + 1 = 6 \end{cases}.$$

Punkti P_6 koordinaadid on:

$$\begin{cases} \xi_6 = (-2) * 1 * 1 \ + 6 * 1 + 1 * 1 + 1 = 6 \\ \eta_6 = 2 * 1 + 5 * 1 + 1 = 8 \end{cases}.$$

Punkti P_7 koordinaadid on:

$$\begin{cases} \xi_7 = (-2) * 1 * 2 \ + 6 * 1 + 1 * 2 + 1 = 5 \\ \eta_7 = 2 * 1 + 5 * 2 + 1 = 13 \end{cases}.$$

Punkti P_8 koordinaadid on:

$$\begin{cases} \xi_8 = (-2) * 0 * 2 \ + 6 * 0 + 1 * 2 + 1 = 3 \\ \eta_8 = 2 * 0 + 5 * 2 + 1 = 11 \end{cases}.$$

Siit näeme, et kaks punkti on eelmise ruudu observatsiooniga kattuvad: $P_3(6,8) = P_6(6,8)$ ja $P_4(2,6) = P_5(2,6)$. Mis on oodatud tulemus, sest ka nende punktide originaalid on samad: $Q_3(1,1) = Q_6(1,1)$ ja $Q_4(0,1) = Q_5(0,1)$.

Nüüd arvutame veel kolme punkti $Q_9(2,0);\ Q_{10}(2,1);\ Q_{11}(2,2)$ koordinaadid.
Punkti P_9 koordinaadiks saame:

$$\begin{cases} \xi_9 = (-2) * 2 * 0 \ + 6 * 2 + 1 * 0 + 1 = 13 \\ \eta_9 = 0 * 2 * 0 \ + 2 * 2 + 5 * 0 + 1 = 5 \end{cases}.$$

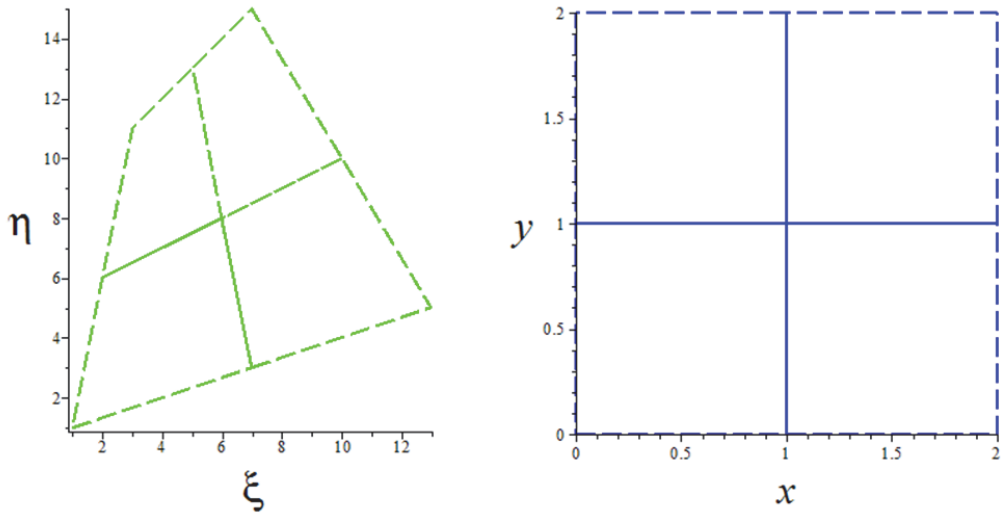
Punkti P_{10} koordinaadiks saame:

$$\begin{cases} \xi_{10} = (-2) * 2 * 1 \ + 6 * 2 + 1 * 1 + 1 = 10 \\ \eta_{10} = 0 * 2 * 1 \ + 2 * 2 + 5 * 1 + 1 = 10 \end{cases}.$$

Ning punkti P_{11} koordinaadiks saame:

$$\begin{cases} \xi_{11} = (-2) * 2 * 2 \ + 6 * 2 + 1 * 2 + 1 = 7 \\ \eta_{11} = 0 * 2 * 2 \ + 2 * 2 + 5 * 2 + 1 = 15 \end{cases}.$$

Juba olemasolevaid punkte ära kasutades saame punktide Q_9, Q_{10} ja Q_{11} abil määrata veel kaks ühikruutu. Olgu esimene neist määratud punktidega (Q_2, Q_9, Q_{10}, Q_3) ja teine $(Q_3, Q_{10}, Q_{11}, Q_7)$. Esitame tulemused joonisel 21, kust on näha, milline on ühikruutude paiknemine observatsiooniteljestikus.



Joonis 21 Parempoolsel joonisel on toodud kalibratsiooniruut. Rakendades sellele teisendust, mis on määratud $\vec{c}, \vec{d}$. Saame tulemuseks vasakpoolsel joonisel esitatud vaatluse tulemuse.

6.2 Bilineaarne teisendus vaatlusest originaali

Arvestades Robotex-i roboti kaamera kalibreerimise ülesannet on eelmises lõigus esitatud teisenduse suund vastupidine ülesande lahendamiseks vajalikule suunale. Leidsime teisenduse, mille abil saame esitada füüsilise maailma punkte observatsiooni teljestikus. See teisendus toimub aga optika seaduspäradest lähtuvalt niikuinii, kui vaatame kalibratsiooniruutu läbi roboti kaamera. Teisendus, mida oleks vaja kaamera kalibreerimise ülesande lahendamiseks, on leitud teisenduse pöördteisendus. Pöördteisenduse saamiseks kasutame eelmisega analoogset konstruktsiooni.

Eeldame, et kogu observatsiooni ja originaali vahel leidub bilineaarne seos ja see seos kehtib kogu pildi ulatuses. Esitame selle seose bilineaarse seose üldkujust lähtuvalt punkti Q_i koordinaadid järgmiselt:

$$\begin{cases} x_i = a_3\xi_i\eta_i + a_2\xi_i + a_1\eta_i + a_0 \\ y_i = b_3\xi_i\eta_i + b_2\xi_i + b_1\eta_i + b_0 \end{cases}.$$

Viitamise lihtsustamiseks organiseerime tundmatud vektoritesse $\vec{a} = (a_3, a_2, a_1, a_0)$ ja $\vec{d} = (b_3, b_2, b_1, b_0)$. Teljed on jätkuvalt tähistatud vastavalt joonisele 20. Olgu originaal-teljestikus ühikruut koordinaatidega $Q_1(0,0)$; $Q_2(1,0)$; $Q_3(1,1)$; $Q_4(0,1)$ ning eeldame, et observatsiooniteljestikus eksisteerivad punktid $P_1(\xi_1, \eta_1)$; $P_2(\xi_2, \eta_2)$; $P_3(\xi_3, \eta_3)$; $P_4(\xi_4, \eta_4)$. Kui punkte $P_1 - P_4$ ei eksisteeriks, siis see tähendaks, et kalibratsioonimuster ei asu kaamera vaateväljas. Avaldame nüüd ühikruudu koordinaadid observatsiooniteljestiku punktide ja bilineaarsete seoste kaudu.

Punkti Q_1 koordinaadid on:

$$\begin{cases} 0 = a_3\xi_1\eta_1 + a_2\xi_1 + a_1\eta_1 + a_0 \\ 0 = b_3\xi_1\eta_1 + b_2\xi_1 + b_1\eta_1 + b_0 \end{cases}.$$

Punkti Q_2 koordinaadid on:

$$\begin{cases} 1 = a_3\xi_2\eta_2 + a_2\xi_2 + a_1\eta_2 + a_0 \\ 0 = b_3\xi_2\eta_2 + b_2\xi_2 + b_1\eta_2 + b_0 \end{cases}.$$

Punkti Q_3 koordinaadid on:

$$\begin{cases} 1 = a_3\xi_3\eta_3 + a_2\xi_3 + a_1\eta_3 + a_0 \\ 1 = b_3\xi_3\eta_3 + b_2\xi_3 + b_1\eta_3 + b_0 \end{cases}.$$

Ning punkti Q_4 koordinaadid on:

$$\begin{cases} 0 = a_3\xi_4\eta_4 + a_2\xi_4 + a_1\eta_4 + a_0 \\ 1 = b_3\xi_4\eta_4 + b_2\xi_4 + b_1\eta_4 + b_0 \end{cases}.$$

Näidislahendus:

Olgu originaalteljestikus ruut koordinaatidega $Q_1(0,0)$; $Q_2(1,0)$; $Q_3(1,1)$; $Q_4(0,1)$ ning observatsiooni teljestikus pilditöötluse poolt tuvastatud ruut koordinaatidega $P_1(1,1)$; $P_2(7,3)$; $P_3(6,8)$; $P_4(2,6)$. Leiame nüüd neid sisendandmeid kasutades teisenduse, mis viiks kaamera pildi koordinaadid vastavusse originaaltelje koordinaatidega. Selleks asendame eelmiseks punktis toodud võrrandites punktide koordinaadid.
Sellest saame võrrandisüsteemi:

$$\begin{cases} a_3 + a_2 + a_1 + a_0 = 0 \\ b_3 + b_2 + b_1 + b_0 = 0 \\ 21a_3 + 7a_2 + 3a_1 + a_0 = 1 \\ 21b_3 + 7b_2 + 3b_1 + b_0 = 0 \\ 48a_3 + 6a_2 + 8a_1 + a_0 = 1 \\ 48b_3 + 6b_2 + 8b_1 + b_0 = 1 \\ 12a_3 + 2a_2 + 6a_1 + a_0 = 0 \\ 12b_3 + 2b_2 + 6b_1 + b_0 = 1 \end{cases}.$$

Võrrandisüsteemi lahendamiseks kasutame taas tarkvarapaketi Maple käsku *solve*. Võrrandisüsteemi lahendiks saame:

$$\begin{cases} a_0 = -\frac{13}{151} \\ a_1 = -\frac{19}{302} \\ a_2 = \frac{20}{151} \\ a_3 = \frac{5}{302} \\ b_0 = -\frac{25}{151} \\ b_1 = \frac{34}{151} \\ b_2 = -\frac{8}{151} \\ b_3 = -\frac{1}{151} \end{cases}.$$

Lähtudes eeldusest, et teisendus on kogu pildi ulatuses sama, olemegi saanud bilineaarse teisenduse, mis seob selle pildi iga observatsiooniteljestikus oleva punkti tema originaaliga. Punktile Q_i vastav teisendus on üldkujul järgmine:

$$\begin{cases} x_i = \frac{5}{302} * \xi_i\eta_i + \frac{20}{151} * \xi_i + (-\frac{19}{302}) * \eta_i + \left(-\frac{13}{151}\right) \\ y_i = -\frac{1}{151} * \xi_i\eta_i + (-\frac{8}{151}) * \xi_i + \frac{34}{151} * \eta_i + (-\frac{25}{151}) \end{cases}.$$

Ehk:

$$\begin{cases} x_i = \frac{5}{302} * \xi_i\eta_i + \frac{20}{151} * \xi_i - \frac{19}{302} * \eta_i - \frac{13}{151} \\ y_i = -\frac{1}{151} * \xi_i\eta_i - \frac{8}{151} * \xi_i + \frac{34}{151} * \eta_i - \frac{25}{151} \end{cases}.$$

6.3 Bilineaarne teisendus mitme kaadri korral

Olles tuvastanud kalibratsiooniruudustiku mitmest järjestikusest kaadrist tahame nendest saadud tingimusi kombineerida. Tingimusi kombineerime, et parandada hinnangut kaameras tekkiva moonutuse kohta. Eeldame, et kaamera-peegel süsteemi geomeetria on kõigi kaadrite korral sama. See tähendab, et robot ei kaldu põranda suhtes ja, et peegel ei liigu kaamera suhtes. Järgnevalt vaatame viisi mitmest kaadrist saadud info sidumiseks.

Esitame kahe originaalteljestiku punkti Q_i ja Q_j koordinaadid üldkujul.

Punkti Q_i koordinaadid on:

$$\begin{cases} x_i = a_3\xi_i\eta_i + a_2\xi_i + a_1\eta_i + a_0 \\ y_i = b_3\xi_i\eta_i + b_2\xi_i + b_1\eta_i + b_0 \end{cases}.$$

Punkti Q_j koordinaadid on:

$$\begin{cases} x_j = a_3\xi_j\eta_j + a_2\xi_j + a_1\eta_j + a_0 \\ y_j = b_3\xi_j\eta_j + b_2\xi_j + b_1\eta_j + b_0 \end{cases}.$$

Pildilt ruutu tuvastades saame teada ka nurkade paiknemise ruudul. Seega teame, et Q_i ja Q_j asuvad samal ruudu küljel. Sellest lähtuvalt saan arvutada koordinaadi muudu x- ja y-telje sihis $\Delta = Q_i - Q_j$.
X-telje sihis on koordinaadi muut:

$$\Delta x = a_3(\xi_i\eta_i - \xi_j\eta_j) + a_2(\xi_i - \xi_j) + a_1(\eta_i - \eta_j).$$

Y-telje sihis on koordinaadi muut:

$$\Delta y = b_3(\xi_i\eta_i - \xi_j\eta_j) + b_2(\xi_i - \xi_j) + b_1(\eta_i - \eta_j).$$

a_0 taandub välja, sest see muutuja on esindatud mõlema punkti juures.

Eelneva põhjal teame, et küljepikkus avaldub järgmiselt:

$$\begin{aligned}\Delta x^2 + \Delta y^2 = {} & a_3^2(\xi_i\eta_i - \xi_j\eta_j)^2 + 2a_3(\xi_i\eta_i - \xi_j\eta_j)a_2(\xi_i - \xi_j) \\ & + 2a_3(\xi_i\eta_i - \xi_j\eta_j)a_1(\eta_i - \eta_j) + a_2^2(\xi_i - \xi_j)^2 + 2a_2(\xi_i - \xi_j)a_1(\eta_i - \eta_j) \\ & + a_1^2(\eta_i - \eta_j)^2 + b_3^2(\xi_i\eta_i - \xi_j\eta_j)^2 + 2b_3(\xi_i\eta_i - \xi_j\eta_j)b_2(\xi_i - \xi_j) \\ & + 2b_3(\xi_i\eta_i - \xi_j\eta_j)b_1(\eta_i - \eta_j) + b_2^2(\xi_i - \xi_j)^2 + 2b_2(\xi_i - \xi_j)b_1(\eta_i - \eta_j) \\ & + b_1^2(\eta_i - \eta_j)^2\end{aligned}$$

Lisaks sellele teame eeldusest, et Q_i ja Q_j asuvad samal ruudu küljel ja et see ruut on ühikruut. Järelikult on küljepikkus 1:

$$\Delta x^2 + \Delta y^2 = 1.$$

Seega kahe kalibratsiooniruudu vaatlemisel saame 8 võrrandit, mis väidavad, et vaadeldud külgede originaalküljed on pikkusega 1. Muutujaid on kokku 6 $\{a_3, a_2, a_1, b_3, b_2, b_1\}$. Järelikult ei lahene võrrandisüsteem üldjuhul täpselt ja võib ka juhtuda, et lahendit ei leidugi.

Kui süsteem laheneks täpselt, siis iga külje pikkuse viga oleks 0:

$$\Delta x^2 + \Delta y^2 - 1 = 0.$$

Paneme tähele, et see on samaväärne tingimusega:

$$(\Delta x^2 + \Delta y^2 - 1)^2 = 0.$$

Seega iga kordajate komplekti puhul saab hinnata vastava külje pikkuse aproksimeerimisviga $\Delta x_i^2 + \Delta y_i^2 - 1 = E_i$, kui Δx ja Δy on külgede vastavate koordinaatide vahed. Nüüd selle asemel, et leida perfektne lahendus $E_i = 0$ iga külje jaoks, võime otsida kordajate komplekti, mis minimiseerib vigade ruutude summa

$$S = \min_{a,b} \sum_{i=1}^{8} E_i^2.$$

Paneme tähele, et kui leidub perfektne lahendus, siis on see ka minimaalne lahend. Muul juhul saame kordajate komplekti, mis teeb keskmiselt kõige vähem moonutusi. Järelikult on see mõistlik viis kõiki küljepikkuste vigasid minimeerida.

6.4 Kiireima laskumise meetod

Kiireima laskumise meetod ehk *gradient descent* on viis funktsiooni lokaalse miinimumi leidmiseks. Tegu on iteratiivse meetodiga. Otsingu käigus liigutakse mingi sammuga gradiendile vastupidises suunas. Samm võib igal iteratsioonil muutuda. Tavaliselt kahaneb. Gradiendi leidmiseks peame võtma osatuletised kõikide parameetrite järgi. Parameetri a_j järgi võetud osatuletis avaldub:

$$\frac{\partial \sum_{i=1}^{8} E_i^2}{\partial a_j} = \sum_{i=1}^{8} \frac{\partial E_i^2}{\partial a_j},$$

kus

$$\sum_{i=1}^{8} 2E_i \frac{\partial E_i}{\partial a_j}.$$

$$\frac{\partial E_i}{\partial a_j} = \frac{\partial \Delta x_i^2}{\partial a_j} + \frac{\partial \Delta y_i^2}{\partial a_j} = 2\Delta x_i \frac{\partial \Delta x_i}{\partial a_j} + 2\Delta y_i \frac{\partial \Delta y_i}{\partial a_j} = 2\Delta x_i \frac{\partial \Delta x_i}{\partial a_j} + 0 = 2\Delta x \frac{\partial \Delta x_i}{\partial a_j}$$

Osatuletised saab Δx_i avaldisest kergesti leida.

Seejärel võtame osatuletise parameetri b_j järgi:

$$\frac{\partial \sum_{i=1}^{8} E_i^2}{\partial b_j} = \sum_{i=1}^{8} \frac{\partial E_i^2}{\partial b_j},$$

kus

$$\sum_{i=1}^{8} 2E_i \frac{\partial E_i}{\partial b_j}.$$

$$\frac{\partial E_i}{\partial b_j} = \frac{\partial \Delta x_i^2}{\partial b_j} + \frac{\partial \Delta y_i^2}{\partial b_j} = 2\Delta x_i \frac{\partial \Delta x_i}{\partial b_j} + 2\Delta y_i \frac{\partial \Delta y_i}{\partial b_j} = 0 + 2\Delta y_i \frac{\partial \Delta y_i}{\partial b_j} = 2\Delta x \frac{\partial \Delta y_i}{\partial b_j}$$

Osatuletised saab Δy_i avaldisest kergesti leida.

Leides osatuletised kõigi parameetrite suhtes saame rakendada kiireima laskumise meetodit võttes suvalise algpunkti, see tähendab parameetrite a_j ja b_j väärtused ning seejärel liikudes gradiendile vastupidises suunas kuni koondumiseni. Funktsioonil võib olla mitu miinimumpunkti ja seetõttu tuleks protsessi korrata ning lõpuks valida parim tulemus.

6.5 Üldine tükeldus

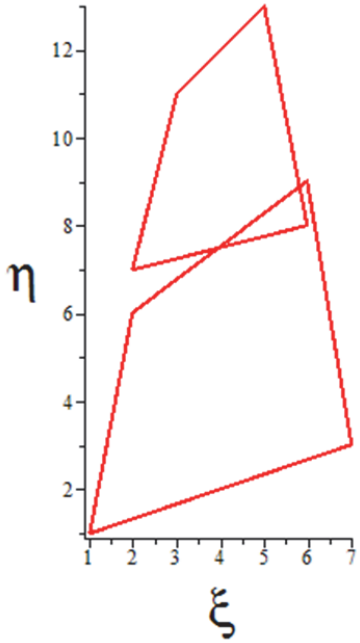
Et lahendada Robotex-i ülesanne, ei piisa kaamera kalibreerimiseks eeldusest, et kogu pildi saab ühe bilineaarse teisendusega siduda originaaliga. Seda seepärast, et ümbrust vaadeldakse hüperboloidpinnaga peegli peegeldusest. Robotex-i ülesande lahendamiseks jagame pildi

osadeks, mille sees eeldame bilineaarset seose piisavust. Näiteks neljaks jagamine toimuks töös kasutatud testandmete põhjal nii, et esimene ristküliku-kujuline jaotis oleks määratud pikslitega mille koordinaadid on (1,1) ja (320,240) - teine (321,0) ja (640,240) - kolmas (0,241) ja (320,480) – neljas (241,321) ja (640,480).

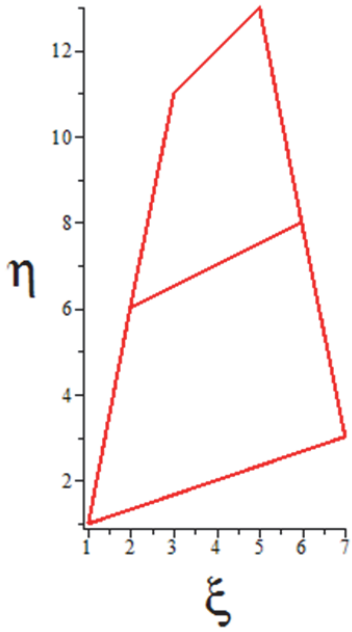
Et vältida olukordi, kus teisendatud alad ei kattu, pean kasutama lisatingimust, et naaberjaotiste ühised nurgad peavad teisenduma samadeks nurkadeks. Joonis 23 sisaldab näidet olukorrast, kus pildi jaotisi on käsitletud eraldiseisvatena ning on tekkinud vastuolud. Joonis 24 sisaldab seevastu näidet olukorrast, kus lisaks jaotiste tingimustele on rakendatud ka ühiste nurkade kattumise tingimust. Bilineaarse teisenduse näidislahenduses langesid nurgad kokku seepärast, et rakendasime kogu pildile sama bilineaarset teisendust. Seega üldise tükelduse korral tuleb lisaks kauguste minimiseerimisele arvestada tingimusega jaotise nurkade kattumise, ehk kooskõlalisuse kohta. See muudab antud ülesande keerukamaks.

Joonisel 22 on esitatud üldine programmi osade järgnevus. Kõigepealt tuleks tuvastada kalibratsioonipunktid. Seejärel konstrueerida nendest kõik tingimused, nii küljepikkuste, kui teisenduste kooskõlalisuse mõttes. Lõpuks tuleks süsteemi lahend minimiseerida.

Joonis 22 Programmi üldine struktuur. Toodud on programmi tegevuste järgnevus. Punktide tuvastamise osa on töös implementeeritud.



Joonis 23 Näide olukorrast kus kahte pildi jaotist on käsitletud eraldiseisvana ja nüüd on pildil osad mille kohta kalibratsioon infot ei sisalda ning osad mille kohta sisaldub vastuoluline info.



Joonis 24 Näide olukorrast kus pildi jaotistele on rakendatud ka tingimust, et ühised nurgad peavad kattuma. Sellise tingimuse lisamine väldib vastuolude teket.

Kahjuks ei jõutud antud töö käigus käesolevas peatükis esitatud lahenduskäiku programmina esitada.

Kokkuvõte

Käesolev töö käsitles ühte tehisnägemise arendamisega seotud probleemidest - kuidas luua seos robotile läbi kaamera laekuva pildi ja reaalse maailma vahel. Seose loomiseks on vajalik kaamera kalibreerimine testpildi põhjal. Uuriti võimalust sellise programmi kirjutamiseks, mis on võimeline genereeritud piltide põhjal esitama teisenduse pildi ja mingi kindla reaalse maailma tasandi vahel. Ülesande lahendamiseks on mitmeid üldisi võimalusi, mis põhinevad enamasti kaamera parameetrite hindamisel. Töös käsitletud lähenemine eeldab, et soovitakse kalibreerida mingi kindla tasandi suhtes ruumis.

Lahenduse üldidee seisnes ühe valge ruudu kasutamises kaamera kalibreerimiseks. Kõigepealt tuli kalibratsiooniruut tuvastada. Seejärel rakendati teadmist, et füüsilises ruumis oli kalibratsioonimuster ruut. Observeeritava ruumi ja füüsilise ruumi vahelise teisenduse leidmisel kasutati bilineaarset seost. Võrrandite rohkuse tõttu rakendati kiireima laskumise meetodit, et minimiseerida kõigi küljepikkuste vigasid. Lahenduskäiku rakendava programmi kirjutamiseni ei jõutud.

Töös kirjeldati mitmeid varemloodud või praegu arenduses olevaid kaamera kalibratsiooni tööriistu. Kirjeldustes toodi välja tööriistade vahelisi seoseid ja eripärasid.
Õpiti kasutama POVRay tarkvarapaketti katseandmete simuleerimiseks, lisaks sellele süvendati programmeerimiskeele Python kohta käivaid teadmisi.
Implementeeriti *proof of consept* tüüpi programm, mis kasutab modifitseeritud versiooni W. Sun [41] poolt kirjeldatud kihilisest nurgatuvastusmeetodist.
Esitati viis pildi kalibreerimiseks vajalikust bilineaarsest teisendusest koos näidislahendusega üksikute kaadrite juhul.

Töö edasiarendamiseks tuleks kirjeldatud kalibratsioonimeetod implementeerida ja seejärel seda vastavalt reaalse maailma katsetulemustele optimeerida.

Calibration of Robot's Omnidirectional Vision System

Bachelor thesis
6 ECTS
Janno Jõgeva

Abstract

The aim of this paper is to present a method for camera calibration. The actual implementation of the calibration process itself is not included in this paper, the solution is only theoretical. The camera calibration is considered in this paper in the meaning of spatial calibration and not in the sense of photometric camera calibration. The calibration problem arose in the robotics competition Robotex. The exact location of objects on a certain plane needs to be estimated by the robot in real-time. This means that there is a need for mapping between pixels and real world distances.

This paper presents a set of existing methods for solving this camera calibration problem.
The test data is generated using POVRay ray-tracing software. This enables predictable test cases for the software. A white square sheet of paper is used as the calibration pattern. A simplified version of W. Sun's [41] corner detection algorithm is implemented to extract the location of the calibration pattern from an image. After the corners have been extracted, a method based on bilinear interpolation is proposed to calibrate the camera. The information that the calibration pattern is a square in the physical world is used in the calibration method. The proposed method suggests that using more frames increases the accuracy of the calibration. In order to improve the accuracy, the image is divided into subsections that are assumed to have a bilinear transformation from the physical world to the observed image.

The next research in this field should implement the suggested method to verify it's accuracy.

Viited

[1] Cool Kitchen Gadgets, „Cool Kitchen Gadgets,“ 22 aprill 2012. [Võrgumaterjal]. Link: http://www.coolkitchengadget.org/wp-content/uploads/2012/04/cctv.jpg. [Kasutatud 06 mai 2012].

[2] Tamron, „CCTV Lenses/Model : 13FM22IR,“ [Võrgumaterjal]. Link: http://www.tamron.co.jp/en/data/cctv_ir/13fm22ir.html. [Kasutatud 06 mai 2012].

[3] M. Avellino, „Lonely Planet Images,“ [Võrgumaterjal]. Link: http://www.lonelyplanetimages.com/search/110307?keywords=mirror. [Kasutatud 06 mai 2012].

[4] J.-Y. Bouguet, „Camera Calibration Toolbox for Matlab,“ Computer Vision Research Group, Dept. of Electrical Engineering, California Institute of Technology, [Võrgumaterjal]. Link: http://www.vision.caltech.edu/bouguetj/calib_doc/. [Kasutatud 04 mai 2012].

[5] Kollektiivne, „OpenCV (Open Source Computer Vision),“ [Võrgumaterjal]. Link: http://www.opencv.org. [Kasutatud 04 mai 2012].

[6] Kollektiivne, „the Vision-something-Libraries,“ [Võrgumaterjal]. Link: http://vxl.sourceforge.net/. [Kasutatud 06 mai 2012].

[7] G. Panin, T. Röder, S. Klose, C. Lenz ja S. Nair, „A general-purpose tracking library,“ [Võrgumaterjal]. Link: http://www.opentl.org. [Kasutatud 06 mai 2012].

[8] D. Scaramuzza, *Omnidirectional Vision: from Calibration to Robot Motion Estimation,* Zürich, 2008.

[9] D. Stoyanov, „Camera Calibration Tools,“ [Võrgumaterjal]. Link: http://www.cs.ucl.ac.uk/staff/Dan.Stoyanov/calib/. [Kasutatud 04 mai 2012].

[10] C. Mei, „Omnidirectional Calibration Toolbox,“ [Võrgumaterjal]. Link: http://homepages.laas.fr/~cmei/index.php/Toolbox. [Kasutatud 04 mai 2012].

[11] V. Vezhnevets, „GML MatLab Camera Calibration Toolbox,“ Graphics and Media Lab CMC department, Lomonosov Moscow State University, [Võrgumaterjal]. Link: http://graphics.cs.msu.ru/en/science/research/calibration/matlab. [Kasutatud 04 mai 2012].

[12] V. Vezhnevets, A. Velizhev, N. Chetverikov ja A. Yakubenko, „GML C++ Camera Calibration Toolbox,“ Graphics and Media Lab CMC department, Lomonosov Moscow State University, [Võrgumaterjal]. Link: http://graphics.cs.msu.ru/en/science/research/calibration/cpp. [Kasutatud 04 mai 2012].

[13] K. Nyholm, „Camera Calibration Toolbox for Octave,“ 15 aprill 2009. [Võrgumaterjal]. Link: http://www.sparetimelabs.com/cameracalib/index.html. [Kasutatud 06 mai 2012].

[14] D. Scaramuzza, „OCamCalib: Omnidirectional Camera Calibration Toolbox for Matlab,“ [Võrgumaterjal]. Link: http://sites.google.com/site/scarabotix/ocamcalib-toolbox. [Kasutatud 04 mai 2012].

[15] J. Barreto, J. Roquette, P. Sturm ja F. Fonseca, „Automatic Camera Calibration Applied to Medical Endoscopy,“ %1 *20th British Machine Vision Conference*, London, 2009.

[16] J. Barreto, J. Roquette, P. Sturm ja F. Fonseca, „EasyCamCalib Software - Easy calibration of a camera using a single image,“ [Võrgumaterjal]. Link: http://arthronav.isr.uc.pt/easycamcalib/. [Kasutatud 04 mai 2012].

[17] J. Heikkilä, „Camera calibration toolbox for Matlab,“ Machine Vision and Media Processing Unit, University of Oulu, [Võrgumaterjal]. Link: http://www.ee.oulu.fi/~jth/calibr/. [Kasutatud 04 mai 2012].

[18] J. Heikkilä, „Geometric Camera Calibration Using Circular Control Points,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence,* kd. 22, nr 10, pp. 1066-1077, oktoober 2000.

[19] T. Svoboda, D. Martinec, T. Pajdla, J.-Y. Bouguet, T. Werner ja O. Chum, „Multi-Camera Self-Calibration,“ [Võrgumaterjal]. Link: http://cmp.felk.cvut.cz/~svoboda/SelfCal/. [Kasutatud 04 mai 2012].

[20] T. Svoboda, D. Martinec ja T. Pajdla, „A convenient multi-camera self-calibration for virtual environments,“ *PRESENCE: Teleoperators and Virtual Environments,* kd. 14, nr 4, pp. 407-422, 2005.

[21] J. Kannala ja S. Brandt, „A Generic Camera Calibration Method for Fish-Eye Lenses,“ %1 *17th International Conference on Pattern Recognition*, Cambridge, 2004.

[22] J. Kannala, „Camera Calibration Toolbox for Generic Lenses,“ 2004. [Võrgumaterjal]. Link: http://www.lce.hut.fi/research/mm/calibration/. [Kasutatud 06 mai 2012].

[23] D. Samper, J. Santolaria, J. J. Aguilar ja et al, „MetroVisionLab Toolbox for Camera Calibration and Simulation,“ [Võrgumaterjal]. Link: http://metrovisionlab.unizar.es/. [Kasutatud 06 mai 2012].

[24] V. Douskos, L. Grammatikopoulos, I. Kalisperakis, G. Karras ja E. Petsa, „FAUCCAL: an open source toolbox for fully automatic camera calibration,“ %1 *XXII CIPA Symposium on Digital Documentation, Interpretation & Presentation of Cultural Heritage*, Kyoto, 2009.

[25] V. Douskos, „FAUCCAL (Fully Automatic Camera Calibration),“ 01 veebruar 2010. [Võrgumaterjal]. Link: http://portal.survey.ntua.gr/main/labs/photo/staff/gkarras/fauccal.html. [Kasutatud 06 mai 2012].

[26] V. Douskos, I. Kalisperakis ja G. Karras, „Automatic calibration of digital cameras using planar chess-board patterns,“ %1 *Optical 3-D Measurement Techniques VIII*, Zürich, 2007.

[27] V. Douskos, I. Kalisperakis, G. Karras ja E. Petsa, „Fully automatic camera calibration using regular planar patterns,“ *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* kd. XXXVIII, pp. 21-26, 2008.

[28] J. P. Barreto, K. Daniilidis, N. Kelshikar ja R. Molana, „Tele-Immersion: EasyCal,“ [Võrgumaterjal]. Link: http://www.cis.upenn.edu/~kostas/tele-immersion/research/downloads/EasyCal/. [Kasutatud 04 mai 2012].

[29] K. H. Strobl, W. Sepp, S. Fuchs, C. Paredes ja K. Arbter, „DLR CalDe and DLR CalLab,“ Institute of Robotics and Mechatronics, German Aerospace Center (DLR), [Võrgumaterjal]. Link: http://www.robotic.dlr.de/callab/. [Kasutatud 04 mai 2012].

[30] J. Davis, „tclcalib - A tool for camera calibration,“ [Võrgumaterjal]. Link: http://users.soe.ucsc.edu/~davis/projects/tclcalib/. [Kasutatud 04 mai 2012].

[31] Optical Metrology Centre, „OMC Product Guide – Camera Calibration Software,“ 2001. [Võrgumaterjal]. Link: http://www.optical-metrology-centre.com/Downloads/Products/ProdGuide_CamCal.pdf. [Kasutatud 06 05 2012].

[32] Optical Metrology Centre, „OMC Camera Calibration Software,“ [Võrgumaterjal]. Link: http://www.optical-metrology-centre.com/products_camera_calibration_software.htm. [Kasutatud 06 05 2012].

[33] Optical Metrology Centre, „OMC Camera Calibration Software,“ [Võrgumaterjal]. Link: http://www.optical-metrology-centre.com/products_camera_calibration_software.htm. [Kasutatud 06 mai 2012].

[34] Optical Metrology Centre, „OMC Product Guide – Camera Calibration Software,“ 2001. [Võrgumaterjal]. Link: http://www.optical-metrology-centre.com/Downloads/Products/ProdGuide_CamCal.pdf. [Kasutatud 06 mai 2012].

[35] Z. Zhang, „Flexible Camera Calibration By Viewing a Plane From Unknown Orientations,“ %1 *International Conference on Computer Vision (ICCV'99)*, Corfu, 1999.

[36] Z. Zhang, „A flexible new technique for camera calibration,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 1330-1334, 2000.

[37] Microsoft, „Microsoft Easy Camera Calibration Tool,“ Microsoft, [Võrgumaterjal]. Link: http://research.microsoft.com/en-us/downloads/7e9de40f-06db-452c-a0f2-4fabb4f20f52/. [Kasutatud 06 mai 2012].

[38] M. Personnaz, „Tele2,“ [Võrgumaterjal]. Link: http://perception.inrialpes.fr/Soft/calibration/index.html. [Kasutatud 06 mai 2012].

[39] M. Personnaz, „Tele2 version 3.1 User's Guide,“ Märts 2002. [Võrgumaterjal]. Link: http://perception.inrialpes.fr/Soft/calibration/Versions/manual31.pdf. [Kasutatud 06 mai 2012].

[40] A. Hornberg, Toim., Handbook of Machine Vision, WILEY-VCH Verlag GmbH & Co. KGaA, 2006.

[41] W. Sun, X. Yang, S. Xiao and W. Hu, “Robust Checkerboard Recognition for Efficient Nonplanar Geometry,” 2008.

[42] V. N. Dao ja M. Sugimoto, „A Robust Recognition Technique for Dense Checkerboard Patterns,“ %1 *International Conference on Pattern Recognition*, Istanbul, 2010.

Lisad

Lisa 1 - POVRay näidisseadistus

```
All_Console=On
Antialias=On
Antialias_Depth=6
Bits_Per_Color=8
Display =off
Height=480
Input_File_Name=sheet.pov
/* The file name is actually generated from input file. This is just to specify the output
folder. */
Output_File_Name=./s01/
/* This is to specify POVRay version. */
Version=3.6
Width=640
/* For image image series:*/
Initial_Frame=1
Final_Frame=10
Initial_Clock=0
Final_Clock=2
/*Sets AA method to 1-adaptive or 2-recursive */
Sampling_Method=1
```

Lisa 2 - POVRay maailma mudeli näidis

```
/*
This programm is meant for easy computer vision test image generation for my Bachelor thesis.
Author: Janno Jõgeva
Date: 03.05.12
*/

#include "colors.inc"
#include "textures.inc"
#include "shapes.inc"

//Two light sources for the extra brightness
light_source{<0,0,8>colorWhiteshadowless}
light_source{<0,0,8>colorWhiteshadowless}

// If only perspective distorion is needed then use this section.
/*
#declare my_pinhole_camera =  camera
 {
   perspective
   location <0, 0, 9> //clock range is given in .ini file
   look_at <0, 5, 0>
   angle 70
 }
camera { my_pinhole_camera }

box { <-3+1.5*clock,3,-0.01>, <0+1.5*clock, 6, 0> pigment { color White } }
*/

#declare my_spherical_mirror_camera =  camera
{
spherical
location<0,0,9>//clock range is given in .ini file
look_at<0,0,0>
angle160
}
camera{my_spherical_mirror_camera}

box{<-10+5*clock,-2.5,-0.01>,<0+5*clock,2.5,0>pigment{colorWhite}}


//If a plane is neede instead of finite size objects use this section.
/*
plane { // xy-plane
   z, 0 // eqvivalent to "<0, 0, 1>, 0"
     //This enables checkerboard pattern.
   pigment {checker color Black color White }
   //translate <1.0*clock, 0, 0> // Fly-by direction can be specified here!
   rotate <0, 0, 360*clock> // Rotational movement can be adjusted here.
}
*/
```

Lisa 3 - Python-is kirjutatud programmi tekst

http://kodu.ut.ee/~janno/baka/bakaJogeva.py