TARTU ÜLIKOOL
Loodus- ja täppisteaduste valdkond
Füüsika instituut

Uku Kert Paidra

Komeedivaatluskaamera OPIC asukoha määramine sellega pildistatud fotode põhjal

Bakalaureusetöö (6 EAP)
Füüsika, keemia ja materjaliteadus

Juhendaja:
Mihkel Pajusalu, PhD

Tartu 2022

Komeedivaatluskaamera OPIC asukoha määramine sellega pildistatud fotode põhjal

Komeedivaatluskaamera OPIC (ingl *Optical Periscopic Imager for Comets*) on Tartu Ülikooli Tartu observatooriumi poolt välja arendatav kaamera, mis läheb Euroopa Kosmoseagentuuri komeedipüüduri retkel osaleva sondi B2 pardale. Retke eesmärk on uurida Öpiku-Oorti vööst värskelt väljunud komeeti, mis alustab teekonda päikesesüsteemi sisemistele aladele. Käesolevas töö eesmärgiks on hinnata piltide põhjal OPIC asukoha määramise täpsust. Selle jaoks loodi komeedi tuumast 3D mudel, kasutades komeedivaatluskaamera CoCa (ingl *Comet Camera*) simuleeritud pilte ning SfM (ingl *Structure from Motion*) ahelat. Koostati kaks algoritmi, mis võrdlevad OPIC simuleeritud pilte saadud mudeli vastu eesmärgiga määrata OPIC asukoht ja pööre komeedituuma suhtes ilma pilditunnuseid kasutamata ning hinnati nende algoritmide täpsust.

Võtmesõnad: 2D-3D sobitamine, masinnägemine, SfM, kaamera asukoha määramine, komeedipüüdur

CERCS: T320 - Kosmosetehnoloogia

OPIC pose estimation using images taken with the camera

The Optical Periscopic Imager for Comets (OPIC) is a periscope camera that is being developed by Tartu University's Tartu Observatory. It is planned to go onboard probe B2, which is a part of the comet interceptor mission developed by the European Space Agency. The mission's goal is to study comets that have exited the Öpik-Oort cloud and have started their way into the inner solar system. The goal of this paper is to estimate the pose of OPIC from the taken images and to determine the error of the estimation. For this purpose, a 3D model of the comet's nucleus is created using the Structure from Motion (SfM) pipeline and images from the Comet Camera (CoCa). In addition two methods were developed to test OPIC images against the 3D model, to determine the pose of the camera without using image features and the methods accuracy was determined.

Keywords: 2D-3D matching, computer vision, SfM, camera pose determination, comet interceptor

CERCS: T320 - Space technology

Sisukord

Sissejuhatus

Komeetide uurimine võimaldab paremini mõista päikesesüsteemi teket, sest nendes leidub jääd, mis on seal olnud päikesesüsteemi algusest peale. Kõik seni uuritud komeedid on sisenenud päikesesüsteemi sisealadele ja tiirelnud lähedalt ümber Päikese. Nende kuju ja koostis on tähe poolt mõjutatud ning info nende tekke kohta võib olla muutunud. Selleks, et teada saada, millised olid komeedid päikesesüsteemi formeerumise ajal, tuleb vaadelda komeete, mis on just väljunud Öpiku-Oorti komeedipilvest ning mida Päike pole veel mõjutanud.[1]

Ülesanne ei ole triviaalne, kuna ei ole teada, millal sobiva trajektooriga komeedid vööst väljuvad. Seepärast on loodud komeedipüüduri retk, mille eesmärgiks on lennata Päikese-Maa Lagrange L2 punkti ja jääda seal ootama väljuvaid komeete. Kui leitakse sobiva trajektooriga komeet, siis alustatakse sellele lähenemist. Leitud komeedist saaks monoliitse satelliidiga teha vaid ühe möödalennu, mis ei ole piisav vajaliku hulga informatsiooni kogumiseks. Seepärast jaguneb satelliit kolmeks osaks: kosmoseaparaat A, sondid B1 ja B2. Nii saab ühe möödalennuga võimalikult palju infot erinevate nurkade alt.[1]

Sondi B2 trajektoori pole pärast eraldumist teada ja seetõttu on raske sellel olevate instrumentide mõõtmisi ruumiliselt paigutada. Üks võimalus sondi B2 trajektoori määramiseks on kasutada selle pardal oleva kaamera OPIC pilte. Kosmoseaparaadi A pardal on komeedivaatluskaamera CoCa, mille eesmärgiks on luua kõrge lahutusvõimega pilte komeedi tuumast. Saadud piltide abil on võimalik luua tuumast 3D mudel. OPIC piltide ja 3D mudelist on võimalik rekonstrueerida kaamera OPIC ja sondi B2 möödalennu asukohad.[1]

Tegemist on 2D-3D sobitamisprobleemiga, mis on masinnägemises tuntud probleem. Selle lahendamiseks on välja pakutud erinevaid algoritme.[2][3] Tavaliselt põhinevad lahendused piltidelt ja 3D objektidelt pilditunnuste leidmisel ja nende omavahelisel sobitamisel. Komeedivaatluskaamera OPIC asukoha määramiseks sellised meetodid ei sobi. Piltidel olev komeedi tuum on väga väike, läbimõõduga 10-30 pikslit, mistõttu pole võimalik pildilt tunnuseid leida. Asukoha määramiseks tuleb kasutada teistsugust meetodit.[4] Käesoleva töö eesmärgiks on luua algoritm, mis ilma pilditunnusteta määraks kaamera asukoha, ja hinnata selle täpsust.

Töö on jaotatud neljaks osaks. Esimeses osas avatakse komeedipüüduri retke ja

komeedivaatluskaamera OPIC tausta. Teises osas selgitatakse lahti meetodis kasutatav teooria. Kolmandas osas kirjeldatakse algoritmi loomise meetoodikat. Neljandas osas hinnatakse loodud algoritmi täpsust.

1 Taust

1.1 Öpiku-Oorti komeedipilv

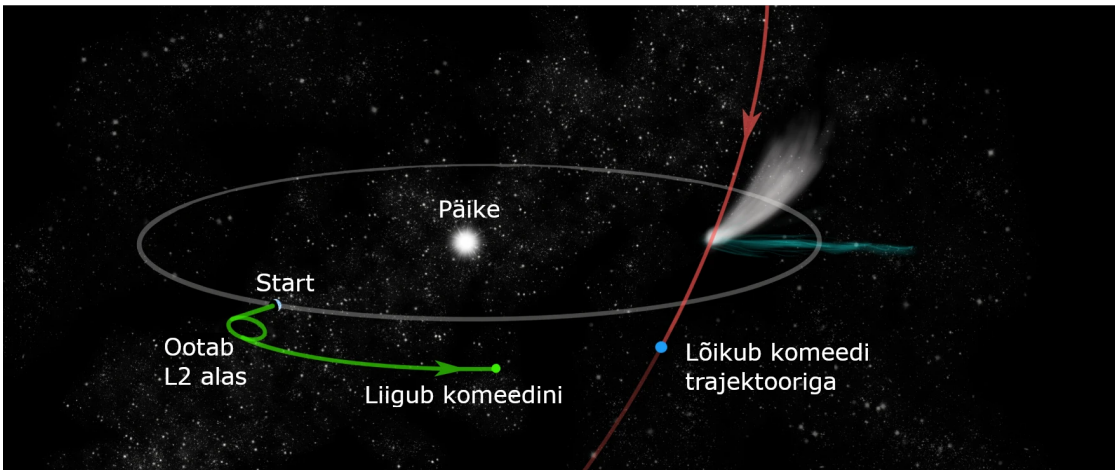
Kääbusplaneedi Pluuto kaugus Päikesest on ligikaudu $6 \cdot 10^9$ km ja Päikese gravitatsiooniline mõjuala ligikaudu $2 \cdot 10^{13}$ km. Nende kahe vahele jääb Öpiku-Oorti komeedipilv, milles on suurusjärgus $10^{12} - 10^{13}$ komeeti.[5] Selle olemasolu võimalikkust tõestasid Ernst Öpiku arvutused aastal 1932 [6], millest komeedipilve idee arendas välja Jan Hendrik Oort [7], kelle järgi vöö nimetuse sai.

Öpiku-Oorti vöös on komeedid miljardeid aastaid vanad ning praktiliselt muutumatud alates nende tekkest. Komeete uurides loodetakse saada informatsiooni päikesesüsteemi algest, kuid uuringute teostamine pole lihtne, sest komeedid asetsevad kaugel ning nendeni jõudmine on keeruline. Aeg-ajalt väljuvad komeedid vööst ja sisenevad Päikese soojussfääri.[8] Soojussfääri sisenedes tekivad komeetidele iseloomulikud sabad, mis koosnevad Päikese mõjul komeedi pinnalt eraldunud materjalist. Esmakordselt päikesesüsteemi sisealadele sisenevaid komeete on keeruline uurida, sest tavaliselt avastatakse komeedid liiga hilja: paar kuud kuni aasta enne periheelist möödumist, pärast mida alustavad komeedid teekonda tagasi välisesse päikesesüsteemi.[1] See on probleem, sest kosmoseretkede planeerimine võtab aastakümneid. Selles ajaskaalas ei ole võimalik avastada komeete ja planeerida uurimiseks kosmoseretki. Kavandatav komeedipüüdur ongi loodud selleks, et päikesesüsteemi sisealadele sisenevatele komeetidele kiiremini reageerida.[1]

1.2 Komeedipüüdur

Komeedipüüduri retke mõte on parkida komeedipüüdur Maa ja Päikese Lagrange L2 punkti. See on ala, kus Maa ja Päikese gravitatsioonilised jõud tasakaalustavad teineteist. Moodustub stabiilne ala, kus komeedipüüdur võib püsida Maast pidevalt samal kaugusel, kasutades minimaalselt kütust. Komeedipüüdurit pargitakse seni, kuni avastatakse uurimiseks sobiv komeet. Seejärel saab saata komeedipüüduri komeeti uurima. Selliselt saavutatakse kiirem reaktsiooniaeg, kui traditsiooniliste retkede puhul võimalik.[1] Komeedipüüduri liikumise faase illustreerib joonis 1.

Komeedipüüdur koosneb kolmest osast: kosmoselaev A, mis käitub emalaevana ning sondid B1 ja B2. Kosmoselaeva A teadusliku instrumentatsiooni hulka kuuluvad CoCa



Joonis 1. Komeedipüüduri retke liikumisfaasid. Komeedipüüdur, mille trajektoor on tähistatud rohelisega, stardib Maalt ja liigub Maa ja Päikese Lagrange L2 alasse ning jääb ootele. Komeedi trajektoor on tähistatud punasega. Komeedi tuvastamisel, liigub komeedipüüdur ootealalt komeediga lõikuvale trajektoorile. Komeedi ja komeedipüüduri kohtumispunkt on tähistatud sinise punktiga. Joonis on illustratiivne ja võetud allikast [1].

ja MIRMIS (intl *Multispectral InfraRed Molecular and Ices Sensor*). Lisaks teaduslikele vaatlusseadmetele on kosmoselaeva A pardal kaugsuhtluseks vajalik instrumentatsioon ja tõukesüsteem.[4]

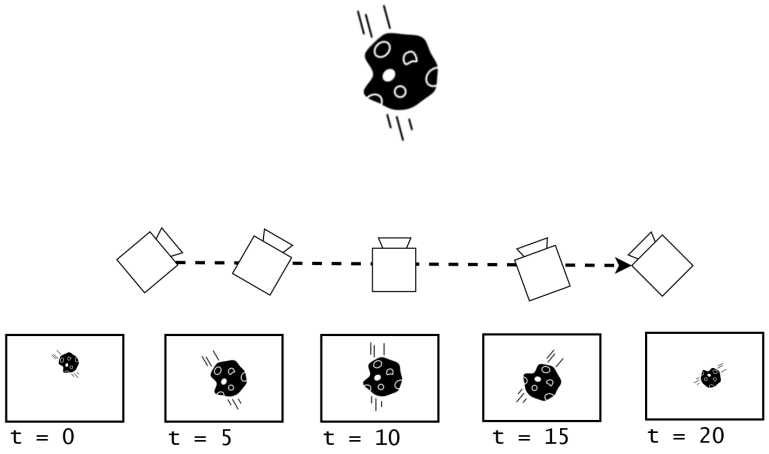
Sondid B1 ja B2 on väiksemad ning on algselt kosmoselaeva A pardal. Sondi B2 pardal on teaduslikud instrumendid OPIC, EnVisS (ingl *Entire Visible Sky* ja DPF-B2 (ingl *Dust Field Plasma*).[4] Kuna sond lendab lähedal komeedituumale, peab ta olema kaitstud ka sellest eralduvast tolmust. Seetõttu on sellele loodud tolmukaitse, mille taha saab peita mehaanilistele häiretele tundliku elektroonika ja optika.[4]

Sobiliku komeedi avastamisel liigub komeedipüüdur komeediga lõikuvale trajektoorile. Jõudes planeeritud kaugusele, toimub protsess, mille käigus sondid B1 ja B2 eralduvad kosmoselaevast A ning alustavad möödalendu komeedist. Kolme möödalennu tulemusel saab komeedi kohta erinevate nurkade alt oluliselt rohkem infot.[1]

Komeedivaatluskaamerad OPIC ja CoCa

Komeedivaatluskaamera OPIC asub sondi B2 pardal. See on Tartu Ülikooli Tartu observatooriumis välja töötatud periskoopkaamera, mille eesmärgiks on pildistada uuritavat komeeti ning seda ümbritsevat tolmupilve. Kaamera vaateväli on 18.3°x18.3° ning kasutatava sensori suurus on 2048x2048 pikslit. See on periskoopkaamera, mis võimaldab paigutada mehaaniliselt tundlikud osad sondi tolmukilbi taha. See vähendab riski, et kõrge impulsiga tolmuosakesed võiksid kaamerat kahjustada. Sondi B2 ja komeedivaatluskaamera OPIC möödalend toimub 400 km kauguselt. Kaamera pöörleb ümber liikumistelje kiirusega vahemikus 4-15 pööret minutis.[4]

Komeedivaatluskaamera CoCa asub kosmoselaeva A pardal ning selle eesmärgiks on saada võimalikult suure lahutusvõimega pilte komeedi pinnast. CoCa väljastab pilte neljas sageduskanalis. Kaamera vaateväli on 0.69°x0.92°. Kaamera kasutatava sensori suurus on 1504x2000 pikslit ning see asub pöörlevate peeglite süsteemis, mille eesmärgiks on kaamerat komeedi tuuma poole keerata.[9] Seda möödalendu illustreerib joonis 2. Kosmoselaeva A ja CoCa möödalend toimub ligikaudu 1000 km kaugusel tuumast.[4]



Joonis 2. Kosmoseaparaat A teeb komeedist möödalennu, et komeedivaatluskaamera CoCa saaks komeeti võimalikult lähedalt erinevate nurkade alt pildistada. Nendest piltidest saab arvutada 3D mudeli nähtavast komeedi pinnast. Joonis on illustratiivne.

2 Teooria

2.1 Kaamera vabadusastmed

Töö käigus on vaja määrata kaamera asukohta ja pööret. Nende kahe kirjeldamiseks saab kasutada kuut vabadusastet, neist kolm nihke ja kolm pöördenurkade jaoks. See tähendab, et igale vabadusastmele tuleb määrata muutuja, mille väärtust töö käigus otsitakse. Käesolevas töös on koordinaatsüsteemi kese komeedi tuum, mis on selles koordinaatsüsteemis paigal.

Asukoha kirjeldamiseks kasutatakse tavaliselt ristkoordinaadistikku, kus vasakule-paremale, üles-alla ja edasi-tagasi liikumiseks kasutatakse vastavalt parameetreid (x, y, z). Käesoleva töö puhul vaadeldakse aga ligikaudu sfäärilist komeedi tuuma erinevate nurkade alt. Seetõttu on mõistlik üle minna sfäärilistele koordinaatidele (ρ, ϕ, θ) kasutades selleks valemeid 1,2 ja 3.[10]

$$\rho = \sqrt{x^2 + y^2 + z^2} \tag{1}$$

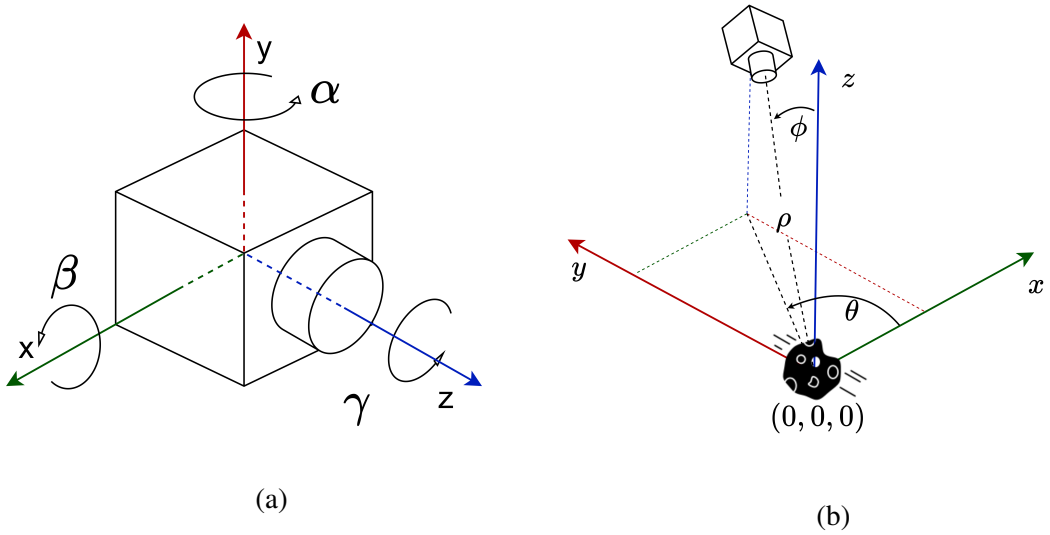
$$\phi = \arccos \frac{z}{\sqrt{x^2 + y^2 + z^2}} \tag{2}$$

$$\tan \theta = \frac{y}{x} \tag{3}$$

Vabadusastmete arv jääb samaks, kuid nende tähendus muutub. Ristkoordinaatsüsteemis on iga parameetri tähenduseks kaugus nullpunktist vastavas teljes. Sfäärilises koordinaatsüsteemis määrab kaugust nullpunktist ainult parameeter ρ. Parameeter ϕ määrab nurka z-telje ning parameeter θ nurka x-telje vahel. Asukohta kirjeldavad sfäärilised parameetrid ja nende tähendused on leitavad joonisel 3b.

Objekti pöörde kirjeldamiseks kasutatakse samuti kolme vabadusastet, mida on näha joonisel 3a. Need on pöördenurk (ingl *yaw*), pikikalle (ingl *pitch*) ja põikikalle (ingl *roll*), mida selles töös tähistatakse muutujatega (α, β, γ). Need on kolm nurka vahemikus $[0; 2\pi)$. Kaamerate puhul määrab γ pööret ümber vaate telje, mis käesoleva töö käigus on z-telg, α ümber ülessuuna (ingl *up vector*) y ning β ümber kõrvaltelje x. Kui anda igale kuuest muutujast $(\rho, \phi, \theta, \alpha, \beta, \gamma)$ mingi väärtus, siis on tulemuseks unikaalne kaamera olek ehk poos. Silmas tuleb pidada, et kõik parameetrid on eraldiseisvad, ühe muutmine

ei mõjuta teisi.



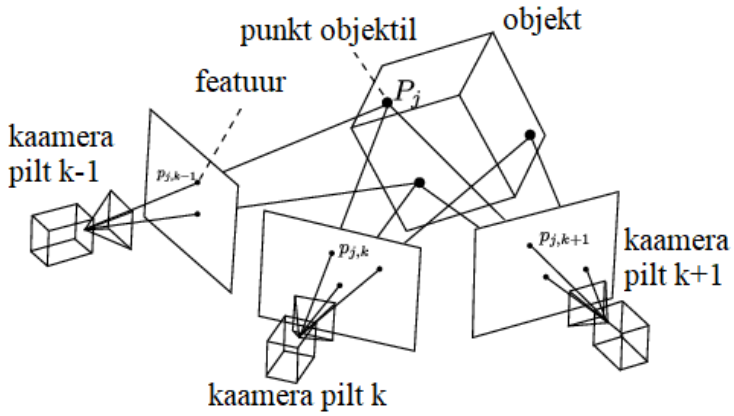
(a) (b)

Joonis 3. Kaamera kuut vabadusastet kirjeldav joonis. Kolm pöörlemist kirjeldavat vabadusastet kaamera taustsüsteemist on näha jooniselt (a). α määrab ära kaamera pöörde ümber y-telje, β ümber x-telje ja γ ümber z-telje . Nende kolme muutujaga on võimalik täielikult ära määrata kaamera pööre 3D süsteemis. Joonisel (b) on näha kaamera kolm asukohta määravat vabadusastet maailma taustsüsteemis, mis on esitatud sfäärilistes koordinaatides ρ, ϕ ja θ. Kaamera kauguse nullpunktist määrab ρ. Nurga x-telje ja kaamera vektori vahel määrab θ ning nurga z-telje ja kaamera vektori vahel määrab ϕ. Need kolm muutujat määravad täielikult ära kaamera asukoha maailma koordinaatsüsteemis. Kolme pöörde ja kolme asukoha muutujaga on võimalik määrata kaamera olek 3D süsteemis.

2.2 Ruumilise struktuuri loomine pildiinfo põhjal

Ruumilise struktuuri loomine pildiinfo põhjal (SfM) on meetod, mis suudab kahest või enamast pildist tekitada piltidel olevast objektist ruumilise kujutise.[11] SfM tuvastab iga pildi kohta tunnused ning sobitab neid teiste piltide tunnustega, mida illustreerib joonis 4. Meetodi tulemusel saadakse 3D mudel objektist ning kaamerate poosid pildistamise hetkel.[12] Meetodiga kaasneb aga puudus: pildil olev ei tohi ajas muutuda. Algoritm ei võta arvesse piltide tegemise vahelisel ajal liikunud osi ja info läheb kaduma.[13]

Meetodi teiseks puuduseks on loodud ruumilise punktipilve absoluutskaala määramatus. Kui pilt on tehtud näiteks kerast, siis sõltumata piltide arvust ei ole võimalik kindlaks teha kera raadiust. Kui stseenis on mitu objekti, siis on võimalik kindlaks teha nende



Joonis 4. SfM meetodi toimimise põhimõte. Joonisel on näha kolme kaameraga tehtud pilte ajahetkedel $k-1, k$ ja $k+1$. Iga pildi pealt leitakse üles mitmeid tunnuseid, näiteks p_j. Nende tunnuste järgi on võimalik määrata tunnusele iseloomulik punkt 3D ruumis. Samuti saab selle järgi teada kaamerate asukohad vastavatel ajahetkedel. Joonis on võetud allikast [12].

mõõtmete suhe. Ruumilise mudeli rekonstruktsiooni suurus on võimalik kindlaks teha, kui kaasata pilti etalon, mille suurus on määratud ja teada. Samuti on võimalik saada mudeli õige skaala, kui on teada kaamera asukohad piltide tegemisel.[14]

Käesolevas töös on sisendandmeteks CoCa tehtud pildiseeria, mis on saadud komeedi tuuma pildistamisest möödalennul. Samuti on teada kaamera asukohad piltide tegemise hetkel. See tähendab, et komeedi tuumast on olemas erinevate nurkade alt tehtud pilte. Kasutades SfM mudelit, on võimalik nendest piltidest luua 3D kujutis komeedi tuumast, mida hiljem tuleb kasutada OPIC asukoha määramisel.

2.3 Kaamera poosi leidmise meetodid

2D piltide ja 3D mudeli punktide järgi kaamera poosi leidmine on masinnägemises tuntud probleem. See leiab rakendust näiteks isejuhtivates autodes, virtuaal- ja liitreaalsuses.[2] Enamus välja pakutud algoritme määravad kaamera poosi pilditunnuste leidmise ja nende sobitamise teel.[2][3][15]

Pilditunnustel põhinevad meetodid

Pilditunnustel põhineva poosi leidmise meetod töötab tavaliselt järgmiselt: pildilt ning 3D mudelilt eristatakse tunnused, need viiakse omavahel vastavusse ning seejärel kasutatakse PnP (ingl *Perspective-n-Point*) lahendajat, mis vastavuses olevate tunnuste järgi leiab kaamera poosi.[3] SfM tarkvaraahel töötab samal põhimõttel, kuid ei vaja sisendiks 3D mudelit. Selle asemel loob algoritm kaamera poosidega koos ka 3D mudeli, millele lisatakse järjest vaateid.[12]

Pilditunnustel põhinevate meetodite eeliseks võrdlusel põhinevate meetodite ees on kiirus.[3] Nende puuduseks on aga sõltuvus pilditunnustest. Meetodite tööks on vajalik piisava arvu pilditunnuste olemasolu. Kui pilditunnuseid ei leidu või neid on vähe, siis ei ole võimalik kaamera poosi määrata. See probleem esineb juhul, kui on kasutada vaid madala detailsusega pildid või kui pildil on ühtlase tooniga objektid.[16]

See tähendab, et pilditunnustel põhinev meetod on eelistatud juhul, kui piltidel on palju varieeruva tekstuuriga objekte. Juhul kui tekstuure ei esine, näiteks ühtlaselt värvitud seinte[17] või puhaste autode puhul[16], siis ei saa pilditunnustel põhinevate meetoditega häid tulemusi. Eelmainitud juhtudel tuleb poosi määrata muude meetoditega, näiteks mõõtmisseadmete[17] või võrdlusel põhineva meetodiga[16].

Võrdlusel põhinev meetod

Käesoleva töö jaoks ei ole pilditunnuseid kasutav lähenemine võimalik, sest varasema töö käigus on leitud, et komeedivaatluskaameraga OPIC tehtud pildid ei ole piisavalt detailsed ja teravad, et pilditunnuseid leida.[4] See tähendab, et on vaja kasutada pilditunnuseid mitte arvestavat meetodit.[4]

Kuna OPIC kaamera sisemised parameetrid on teada, siis on võimalik simuleerida kaamerat 3D keskkonnas. Seega saab teisendada 3D objekti pildiks, kui anda kaamerale poos. Saadud pilt on väga sarnane reaalsele pildile, mida OPIC oleks võimeline samast poosist tegema. Järelikult, kui leida kaamera poos, kus simuleeritud pilt vastab kaameraga OPIC tehtud pildile, siis vastab simuleeritud kaamera poos reaalsele kaamera poosile.

Kuna kaamera poos koosneb kuuest erinevast vabadusastmest, siis on vajalik koostada funktsioon, mis võtab sisendiks 6 parameetrit, sellega defineerides kaamera oleku 3D ruumis ning väljastab pildi vastavalt nendele parameetritele. Seda pilti on võimalik võrrelda komeedivaatluskaamera OPIC reaalse pildiga ning hinnata tulemust piltide

vahega. Piltide võrdlemiseks kasutatakse keskmist ruuterinevuse funktsiooni (MSE ingl *mean squared error*), mida kirjeldab võrrand 4.

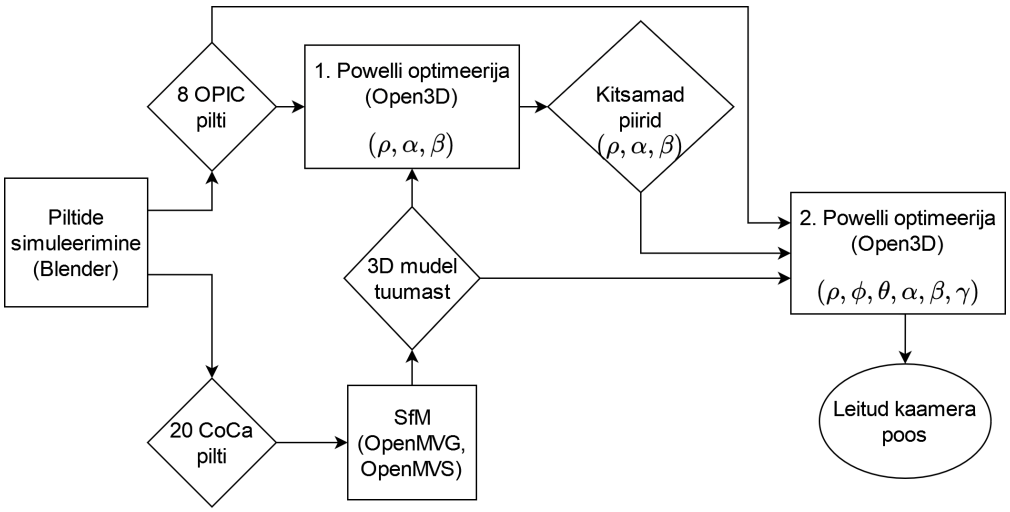
$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{4}$$

Simuleeritud pilt I ja OPIC pilt K lahutatakse teineteisest elementhaaval, nende vahe võetakse ruutu ning summeeritakse üle kõigi pikslite. Sellest summast võetakse keskmine, et saada piltide sarnasust defineeriv skalaarväärtus. Mida väiksem on väärtus, seda sarnasemad pildid teineteisele on. Kui funktsiooni väärtuseks on 0, siis on pildid ideaalselt sarnased.

Kõikide parameetrite kombinatsioonide läbiproovimine võtaks liiga palju aega, mistõttu kasutatakse optimeerimisalgoritmi. Selle töö jaoks on valitud Powelli optimeerimisalgoritm, sest see on efektiivne ning ei vaja lahendamiseks funktsiooni tuletiste olemasolu.[18] Kuna poosi määratakse virtuaalsele kaamerale asukoha andmise ja sealt pildistamisega ning piltide võrdlusega, puudub meetodil analüütiline vorm ning tuletiste võtmine on keeruline kui mitte võimatu. Powelli algoritm ei vaja tuletisi, mistõttu on see käesoleva töö jaoks sobilik. Algoritm proovib läbi erinevaid muutujate kombinatsioone, liikudes iteratiivselt lokaalse miinimumi poole.

3 Metoodika

Töö käigus simuleeritakse OPIC ja CoCa möödalennul tehtavad pildid kasutades tarkvarapaketti Blender[19]. Saadud CoCa piltidest koostatakse SfM tarkvaraahela abil 3D mudel, millega võrreldakse kaameraga OPIC tehtud pilte, eesmärgiga määrata pildile vastav OPIC poos. Selle jaoks loodi kaks algoritmi: lokaalne ja globaalne optimeerija. Globaalse optimeerija puhul puudub kaamera OPIC ja sondi B2 asukoha kohta info ning ülesandeks on määrata OPIC poos täiesti tundmatus olukorras. Lokaalse optimeerja puhul on ligikaudne OPIC asukoht ja pööre teada ning eesmärgiks on kaamera poosi täpsustada. Töö käiku illustreerib joonis 5.



Joonis 5. Töös loodavate algoritmide ülevaatlik plokkskeem. Esmalt simuleeritakse Blenderis komeedi tuumast möödalend CoCa ja OPIC kaameratega. Seejärel sisestatakse CoCa pildid SfM ahelasse, mille tulemusena saadakse 3D mudel tuumast. Mudel ning OPIC pildid antakse sisendiks esimesele Powelli optimeerijale, mis piirab muutujaid ρ, α ja β. Seejärel sisestatakse OPIC pildid, 3D mudel ning kitsamad piirid teise Powelli optimeerijasse ning saadakse vastuseks otsitav kaamera poos.

3.1 Piltide simuleerimine

Katseandmete jaoks tekitatakse 3D mudel komeedi tuumast läbimõõduga 12 km, millest simuleerida möödalende komeedivaatluskaameratega. Selle jaoks kasutati tarkvaraga

Blender koostatud mudelit, mis vastaks võimalikult hästi reaalsele komeedi tuumale.[4] Blenderi 3D keskkonnas tekitati kaks virtuaalset kaamerat, mis jäljendasid OPIC ja CoCa sisemisi parameetreid. Simuleeritud pilte kasutatakse nii globaalses kui ka lokaalses optimeerijas.

CoCa pildid

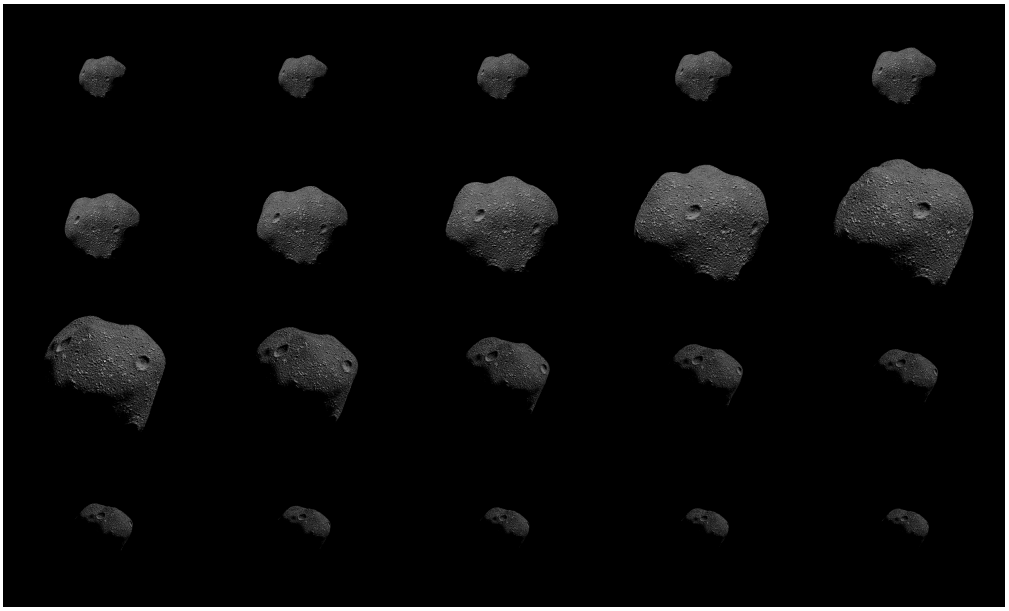
Esmalt simuleeriti CoCa möödalend, mille trajektoori lähim punkt komeedi tuuma keskosast on 1000 km. Möödalend toimub komeedi valgustatud osast. Virtuaalsel möödalennul tegi CoCa 20 pilti tuuma jälgides, pöörates end pidevalt komeedi poole. Simuleeritud pilte on näha joonisel 6.

Lisaks piltidele on teada ka CoCa kiirus, mis saadakse kosmoseaparaadi A mõõteriistade abil. Kuna on teada CoCa kiirus, siis on võimalik arvutada vahemaa piltide vahel. See informatsioon on oluline, et mitmevaatelise geomeetria abil koostada komeedi tuum absoluutskaalas. Seda teadmist jäljendati Blenderis ja koos piltidega väljastati kaamera koordinaadid piltide tegemise ajal.

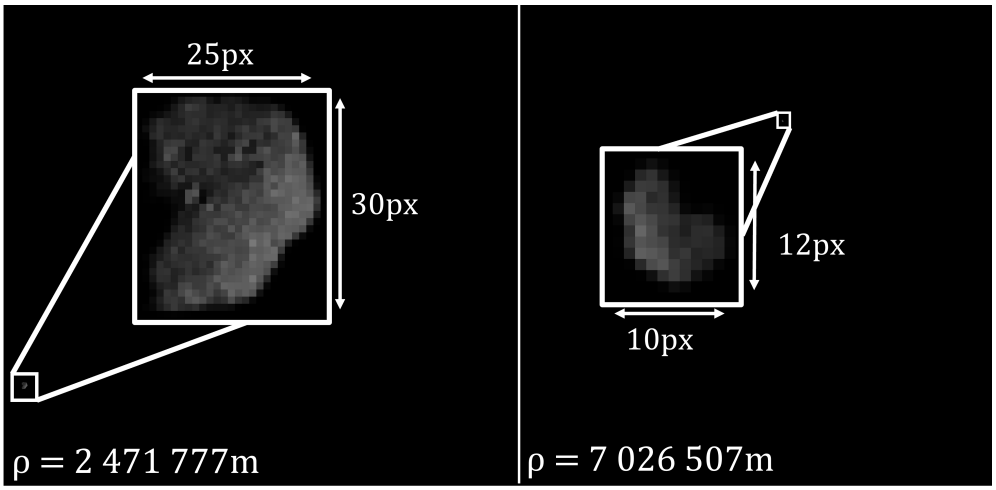
OPIC pildid

Blenderis simuleeriti ka komeedivaatluskaamera OPIC pildid. Lähim asukoht komeedi tuuma ja komeedivaatluskaamera OPIC vahel on 400 km. Sondi B2 diameeter on ligikaudu 1 m ning ta pöörleb liikumissuunas. Kahte simuleeritud OPIC pilti on näha joonisel 7. Vasakpoolne pilt tehti simuleeritud OPIC kaameraga, kui selle kaugus komeedist oli vähim, mille puhul komeedi tuum jäi veel kaadrisse. Komeedi läbimõõt sellel pildil on vahemikus 25-30 pikslit ja kaamera kaugus komeedist 2471.777 km. Parempoolsel pildil on näha komeedi tuuma kaugusel 7026.507 km, kus on komeedi tuuma läbimõõt pildil vahemikus 10-12 pikslit. Nendelt kaadritelt ilmneb, miks traditsioonilised 2D-3D sobitamise algoritmid sellel juhul ei töötaks - infot on liiga vähe, et piisavalt tunnuseid leida.

Eelmises lõigus mainitud OPIC möödalennu parameetrid on planeeritavad. Tegelikkuses toimub möödalend alles aastate pärast, mistõttu olukord võib muutuda. Selleks, et loodav sobitusalgoritm oleks universaalsem, simuleeriti lisapilte komeedi tuuma läbimõõduga 5-60 pikslit. See võimaldab algoritmi üldisemalt hinnata ning teha laialdasemaid järeldusi.



Joonis 6. Komeedivaatluskaamera CoCa Blenderis simuleeritud pildid. Möödalennul pöörab kaamera end komeedi tuuma poole, et saada infot komeedi kuju kohta võimalikult paljude nurkade alt. Kaamera vaateväli on väike, mis võimaldab ka 1000 km pealt saada hea lahutusvõimega pilte.



Joonis 7. Blenderis simuleeritud komeedivaatluskaamera OPIC tehtud pildid. Vasakul on näha komeedi tuuma kaugusel 2471 km ja vasakul 7026 km. Pildil on suurendatud komeedi tuumasid ligikaudu 30 korda, et seda oleks paremini näha. Komeedi tuum on kaamera simuleeritavatel piltidel läbimõõduga vahemikus 10-30 pikslit.

3.2 Virtuaalse 3D stseeni ehitamine

Komeedi tuuma 3D mudeli koostamine

Piltide järgi 3D mudeli koostamiseks kasutatakse SfM tarkvaraahelat. Tarkvarapakette, millega sooritada SfM ahelat on mitmeid, kuid selle töö käigus kasutatakse tarkvarapakette OpenMVG (ingl *open multi view geometry*)[20] ja OpenMVS (ingl *open multi view stereometry*)[21]. Neid otsustati töö jaoks kasutada, sest mõlemad on vabavaralised ja autoril on nendega varasem kogemus. Täpsem kirjeldus OpenMVG ja OpenMVG tarkvaraahelast on leitav allikast [22].

Simuleeritud CoCa pildid koos asukohtadega sisestati SfM ahelasse, mis väljastas komeedi tuumast 3D mudeli, mida on näha joonisel 8. See mudel sisaldab infot, mis on saadud CoCa piltidelt. Kuna möödalend toimub komeedi ühelt poolelt, siis tagumist külge mudelil pole, nagu jooniselt näha. See ei ole OPIC asukoha määramisel probleem, sest OPIC ja CoCa mööduvad komeedist sama külje poolt, mistõttu tagumine külg ei jää OPIC kaadrisse ning ei mõjuta simuleeritud kaadri kvaliteeti.

Joonis 8. CoCa piltidest SfM ahela poolt koostatud 3D mudel kuue erineva nurga alt. Kuna CoCa teeb möödalennu ainult komeedi ühest küljest, siis on ka mudel ligikaudu pool päris tuumast.

Mudeli stseeni paigutamine

Saadud mudel simuleeriti Python 3.10 [23] tarkvarapaketiga Open3D [24]. Tekitatud stseenis on võimalik määrata nii kaamera sisemisi kui ka välimisi parameetreid. Kuna vaadeldakse komeedi tuuma, mis on ligikaudu sfääriline, siis on kasulik üle minna sfäärilise koordinaatsüsteemi peale (ρ, ϕ, θ). See võimaldab intuitiivsemat arusaama simulatsioonist ja lihtsamat muutujaruumi piiramist.

Selleks, et komeedi tuum oleks alati kaadris, kasutati Open3D *lookat()* funktsiooni. Selle funktsiooni parameetriteks on kaamera asukoht, vaatepunkt ja ülemine telg. Kaamera asukoha määravad ära (ρ, ϕ, θ) ning kuna komeedi tuum on staatiline ning see peab pidevalt kaadris olema, siis vaatepunktiks on $(0, 0, 0)$, kus asub komeedi tuuma keskosa. Ülemine vektor näitab, mis suunas on kaadri ülemine osa ja see defineerib γ vabadusastme.

Kasutades programme Blender, OpenMVG ning Open3D, tuleb silmas pidada, et nendes on kasutusel erinevad koordinaatsüsteemid. Näiteks tähistab Blenderis ülemist telge y-telg, kuid OpenMVG ja Open3D programmides on selleks z-telg. Samuti on programmides erinevalt defineeritud kaamera pöördemaatriksid. Blenderis on kaamera vaatevektor defineeritud kui vektor maailma koordinaatide keskmest kaamerani, kuid OpenMVG ja Open3D defineerivad vaatevektorit kui vektorit kaamerast maailma keskpunkti. Samuti on Blenderis pöördemaatriks defineeritud veergude kaupa, kuid teistes programmides ridade kaupa. See tähendab, et Blenderist kaamera pöördemaatriksit väljastades tuleb see enne programmi OpenMVG sisestamist teisendada.

3.3 Globaalne optimeerija

Globaalse optimeerimise puhul ei ole kaamera asukoha ja pöörde kohta mitte midagi teada. Piiramata on optimeerija vabadusastete piirid näha tabelis 1. Need ei ole optimaalsed, sest liiga laiade piiridega otsib optimeerija miinimumi ka nendest kohtadest, kus komeedi tuuma kaadris ei ole. Lisaks pärsib see optimeerija efektiivsust, sest muutujaruumi piiramata jätmisel on rohkem lokaalseid miinimume, kuhu optimeerija valesti koonduda võib. Seetõttu on muutujaruumi piiramine ülimalt tähtis, et suurendada optimeerija kiirust ja täpsust.

Tabel 1. Muutujaruumi esialgsed piirid.

Vabadusaste	ρ (km)	ϕ (rad)	θ (rad)	γ (rad)	α (rad)	β (rad)
Piirid	$(0, \infty)$	$[0, \pi]$	$[-\pi, \pi]$	$[0, 2\pi)$	$[-\pi, \pi)$	$[-\pi, \pi)$

Passiivne piiramine

Muutujaruumi piiramiseks on vaja kasutada lisainformatsiooni, mida on võimalik saada retke tingimustest. Selle jaoks tuleb läheneda erinevalt igale kuuest parameetrist. Kaugust kaamera ja komeedi tuuma keskosa vahel tähistab ρ. Ülesande parameetritest on teada, et OPIC ei ole komeedi tuumale kunagi lähemal kui 400 km, mistõttu saab selle seada alumiseks piiriks. Samuti saab ülemise piiri lähemale tuua simulatsioonide abil. Komeedi tuuma läbimõõt 13000 km kaugusel on viis pikslit, mille järgi saab paika panna ρ ülemise piiri.

Muutujate ϕ ja θ piiramiseks on võimalik kasutada infokildu, et nii OPIC kui ka CoCa lendavad komeedist mööda sama külje pealt. See tähendab, et CoCa ja OPIC näevad sama külge. See on kasulik informatsioon, sest 3D mudel on tehtud CoCa piltidest ja sellel mudelil on ainult need küljed, mis on jäänud CoCa kaadrisse, nagu on näha joonisel 8. See tähendab, et kui arvutada komeedi tuuma mudeli kõikide külgede pinnanormaalid, siis jääb OPIC alati nende normaalide vahele. Kasutades pinnanormaale, on võimalik vältida miinimumi otsimist komeedi tagant, kus see kindlasti ei ole.

Kuna OPIC pöörleb ümber enda telje 360°, siis vabadusastet γ ei saa piiritleda. Muutujaid α ja β saab piiritleda, sest vaatleme ainult pilte, kus komeedi tuum jäi kaadrisse. Kaadrisse jääb komeedi tuum juhul, kui see on kaamera OPIC vaateväljas, milleks on 18.3° mõlemal teljel. Kuna kaadri algasendis on komeedi tuum kaadri keskel, siis nii α kui ka β tähistavad nurga erinevust nullpunkti vaatest. Selleks, et komeedituum jääks kaadrisse võib α ja β väärtuseks olla maksimaalselt pool OPIC vaateväljast, mille arvutuskäik on näha võrrandist 5. Uued piirid pärast passiivset piiramist on leitavad tabelis 2.

$$\alpha, \beta = \pm\frac{FOV_{OPIC}}{2} = \pm\frac{18.3° \cdot \pi}{2 \cdot 180°} \approx \pm 0.16 rad \tag{5}$$

Tabel 2. Muutujaruumi piirid pärast passiivset piiramist.

Vabadusaste	ρ (km)	ϕ (rad)	θ (rad)
Piirid	$[400, 13000]$	sõltub mudelist	sõltub mudelist
Vabadusaste	γ (rad)	α (rad)	β (rad)
Piirid	$[0, 2\pi)$	$[-0.16, 0.16]$	$[-0.16, 0.16]$

Aktiivne muutujaruumi piiramine ehk eelsobitamine

Selleks, et täpsemalt muutujaruumi piire kitsendada, saab kasutada lisainfot, mis on leitav OPIC kaameraga tehtud piltidelt. Pildilt on võimalik välja lugeda, kus tuuma kaadris asub ning mitu pikslit on selle pindala. Viimasega on võimalik kitsendada ρ vabadusastet ning kaadri asukohaga muutujaid α ja β. Nende leidmiseks tuleb arvutada komeedi tuuma massikeskme pikslite koordinaadid. Seejärel simuleeritakse komeedi tuuma kogu lubatud α ja β ulatuses, võetakse nende piltide massikeskmete koordinaadid ning leitakse nende vahel seos.

Kui on leitud ligikaudsed α ja β väärtused, siis on võimalik kitsendada ka ρ piire, sest see on ainuke muutuja, mis tugevalt mõjutab komeedi tuuma suurust kaadris. Selle jaoks loodi eelsobitaja, mis arvutab välja esialgsed α ja β väärtused ning lisaks nendele ka ρ kitsamad piirid kasutades kolme vabadusastmega (ρ, α, β) Powelli optimeerijat, mille veafunktsiooniks on piltide ruutkeskmine erinevus ja mida kirjeldab valem 4.

3.4 Lokaalne optimeerija

Lokaalse optimeerija puhul on teada kaamera ligikaudne asukoht ja pööre ning optimeerija eesmärgiks on seda hinnangut täpsustada. Reaalses olukorras saab hinnangulise poosi teada satelliidi sisemiste mõõteriistade abil. Simulatsioonikeskkonnas on võimalik seda jäljendada, kui väljastada Blenderist piltidele vastavad kaamera poosid.

Blenderist saadud poosid on absoluutselt täpsed. Selleks, et saada hinnangulised poosid, tuleb lisada juhuslik müra. Käesolevas alapeatükis kirjeldatakse asukoha tehete teostamist ristkoordinaatsüsteemis, sest sellisel viisil on lihtsam poosidele müra lisada. Hiljem piiride arvutamisel teisendatakse ristkoordinaatsüsteem sfääriliseks koordinaatsüsteemiks. On teada, et asukoha müra suurusjärk on defineeritav valemi 6 abil, kus $|\Delta C|$ tähistab müravektori pikkust, ρ on OPIC kaugus komeedi tuumast ja n_{px} on pildil oleva komeedi tuuma pikslite arv.

$$|\Delta C| = \frac{\rho}{n_{px}} \tag{6}$$

Hinnangulise asukoha C' saamiseks liidetakse Blenderist saadud asukohale C leitud $|\Delta C|$ ja juhusliku ühikvektori $\hat{p}$ korrutis. Seda kirjeldab valem 7.

$$C' = C + |\Delta C| \cdot \hat{p} \tag{7}$$

Pöörde müravektori pikkuse $|\Delta R|$ leidmiseks tuleb kasutada valemit 8, kus n_{px} tähistab pildil oleva komeedi tuuma pikslite arvu.

$$|\Delta R| = \frac{\pi}{n_{px}} \tag{8}$$

Pöörde müra leidmiseks arvutatakse juhuslik ühikvektor, mis korrutatakse pöörde müravektori pikkusega. Saadud korrutist tõlgendatakse Rodriguese vektorina, mis tähendab, et pöörde suuna määrab vektori suund parema käe reegli järgi ning pöördenurga määrab selle vektori pikkus. Saadud vektor teisendatakse pöördemaatriksiks R_1 ning korrutatakse Blenderist saadud pöördemaatriksiga R. Seda kirjeldab valem 9. Tulemuseks on juhuslik pöördemaatriks R', mis erineb absoluutsest pöördemaatriksist $|\Delta R|$ rad võrra.

$$R' = R \cdot R_1 \tag{9}$$

Lokaalse optimeerija muutujaruumi piirid

Lokaalse optimeerija muutujaruumi piirid on määratud eelmises alapeatükis leitud vigade suurustega $|\Delta C|$ ja $|\Delta R|$. Asukoha piiride arvutamisel on teada müravektori pikkus $|\Delta C|$, kuid ei ole teada, millisele suunale see lisati. Selleks, et otsitav miinimum jääks kindlasti piiride vahele, tuleb eeldada, et viga on maksimaalne igas suunas. Selliselt oleksid asukoha (x, y, z) piirid leitavad valemite 10, 11 ja 12 abil, kus asukoha hinnangu C' koordinaadid on tähistatud kui (x', y', z'). Sfäärilistesse koordinaatidesse teisendatult on piirid leitavad valemite 13, 14 ja 15 abil.

$$[x_{min}, x_{max}] = [x' - |\Delta C|, x' + |\Delta C|] \tag{10}$$

$$[y_{min}, y_{max}] = [y' - |\Delta C|, y' + |\Delta C|] \tag{11}$$

$$[z_{min}, z_{max}] = [z' - |\Delta C|, z' + |\Delta C|] \tag{12}$$

$$[\rho_{min}, \rho_{max}] = \left[\sqrt{x_{min}^2 + y_{min}^2 + z_{min}^2}, \sqrt{x_{max}^2 + y_{max}^2 + z_{max}^2}\right] \tag{13}$$

$$[\phi_{min}, \phi_{max}] = \left[\arccos\left(\frac{z_{max}}{\rho_{min}}\right), \arccos\left(\frac{z_{min}}{\rho_{min}}\right)\right] \tag{14}$$

$$[\theta_{min}, \theta_{max}] = \left[\arctan\left(\frac{y_{min}}{x_{max}}\right), \arctan\left(\frac{y_{min}}{x_{min}}\right)\right] \tag{15}$$

Pöördemuutujate piiride leidmisel on teada pöördenurk $|\Delta R|$, kuid ei ole teada, mis suunas pööre tehti. Piiride leidmiseks kasutatakse sama lähenemist nagu asukoha piiride leidmisel, kus eeldatakse, et viga on igas suunas maksimaalne. Seda eeldades jääb otsitav miinimum kindlasti piiride vahele. Pöörde muutujate (α, β, γ) piirid on leitavad valemite 16, 17 ja 18 abil, kus hinnangulise pöördemaatriksi R' algpakkumised on lahti kirjutatud muutujatena $(\alpha', \beta', \gamma')$.

$$[\alpha_{min}, \alpha_{max}] = [\alpha' - |\Delta R|, \alpha' + |\Delta R]\!] \tag{16}$$

$$[\beta_{min}, \beta_{max}] = [\beta' - |\Delta R|, \beta' + |\Delta R]\!] \tag{17}$$

$$[\gamma_{min}, \gamma_{max}] = [\gamma' - |\Delta R|, \gamma' + |\Delta R]\!] \tag{18}$$

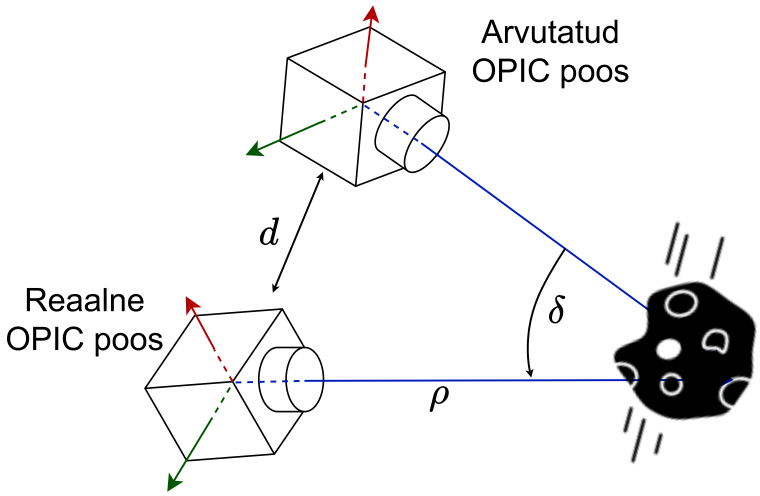
3.5 Asukoha määramine

Lokaalse ja globaalse kaamera poosi leidmiseks kasutati Powelli optimeerimisalgoritmi. Muutujate piirideks seati eelmistes peatükkides leitud piirid ning algpakkumine valiti juhuslikult nende piiride seast. Kuna Powelli algoritm leiab üles vaid lokaalseid miinimume, siis on vajalik jooksutada mitmeid optimeerijaid, et suurendada globaalsesse miinimumi koondumise tõenäosust. Selle jaoks parallelliseeriti probleem, et korraga

saaks ühe protsessori peal toimuda mitu optimeerijat. Kõikidest koondunud tulemustest võetakse vastuseks kõige väiksema veafunktsiooniga poos. Veafunktsiooniks on simuleeritud pildi ja OPIC pildi ruutkeskmine erinevus, nagu on kirjeldatud valemiga 4.

Kuna algandmed on simuleeritud, siis reaalselt on teada pildistamise hetkel kaamera poos, mis on saadud Blenderi simulatsioonikeskkonnast. Seda infot saab kasutada tulemuse täpsuse hindamisel, kuid mitte tulemuse leidmisel, kuna reaalse retke ajal seda infot teada ei ole. Hinnatakse nii pöörde kui asukoha täpsust, mida illustreerib joonis 9. Pöörde täpsus saadakse kaamera pöördemaatriksite nurga absoluutse erinevusega. See näitab, kui suure pöörde peaks kaamera tegema, et jõuda leitud pöördeasendist reaalsesse pöördeasendisse, joonisel tähistab seda δ. Asukoha viga ΔC hinnatakse suhteliselt, jagades asukohtade vahe d kaamera kaugusega komeedi tuuma massikeskmest ρ. Seda kirjeldab valem 19.

$$\Delta C = \frac{d}{\rho} \cdot 100\% \tag{19}$$



Joonis 9. Asukoha vea ja nurga vea määramine. Joonisel tähistab d reaalse ja arvutatud OPIC poosi vahelist kaugust, ρ reaalse poosi kaugust komeedi tuumast ning δ tähistab kahe poosi pöördemaatriksite vahelist nurka. Kauguse viga hinnatakse suhteliselt, jagades kauguse erinevuse d läbi kaugusega tuumast ρ. Nurga vea hinnang on absoluutne ning selle määrab δ.

4 Tulemused

4.1 Globaalse optimeerija tulemused

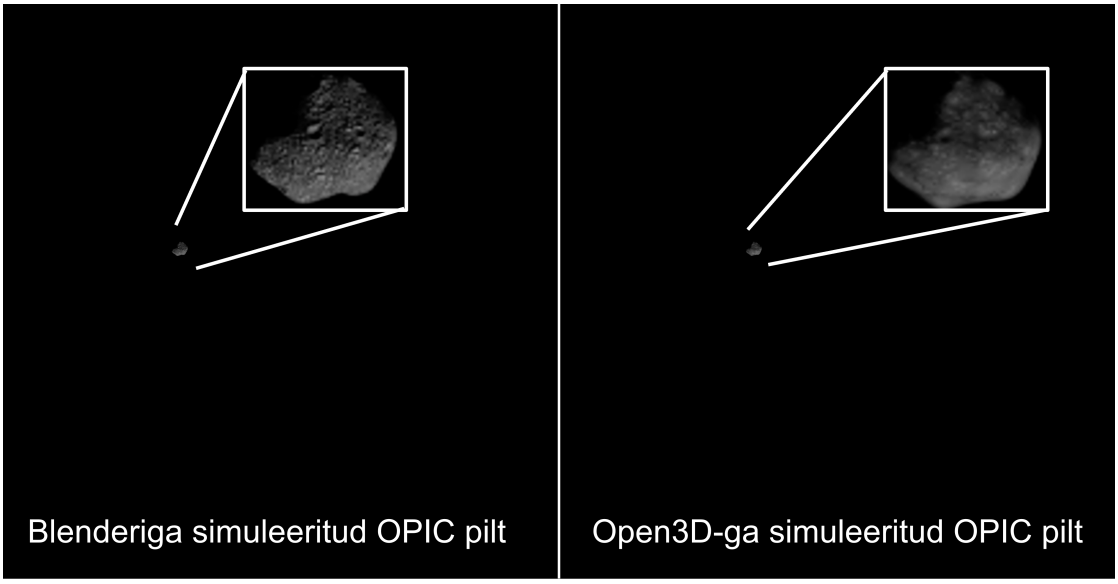
Eelmises peatükis välja pakutud globaalse optimeerimise algoritmi jooksutati 192 korda 8 erineva pildi peal, millel on komeedi tuuma pindalad vahemikus 42-1986 px. Kokku tehti 1536 optimeeringut ja 8 poosi globaalseks optimeerimiseks kulus 18 tundi, kasutades 8 tuuma ja 16 lõimega Intel Core i7 11800H protsessoriga töömasinat, millel on 32GB DDR4 muutmälu. Piltide parameetrid ja optimeeringu tulemused on leitavad tabelist 3.

Tabel 3. Simuleeritud piltide andmed koos optimeerimisalgoritmi tulemustega.

Nr	ρ (km)	Läbimõõt (px)	Pindala (px)	Nurga viga ($^\circ$)	Kauguse viga ($\%$)
1	11759	6	42	21.21	29.32
2	6559	11	106	16.39	21.82
3	4289	16.5	217	117.55	158.26
4	3606	19.5	299	159.22	163.40
5	2985	23.5	432	67.28	111.90
6	1991	33	949	28.12	42.45
7	1650	43.5	1360	30.78	47.62
8	1359	54	1986	13.47	14.10

Tabelist on näha, et tulemused jaotuvad vea suuruse järgi kahte rühma - väikese ja suure veaga tulemused. Viga peetakse väikeseks, kui nurga erinevus on alla 45° ning kauguse viga alla 50%. Sellest väiksemate vigadega tulemust loetakse edukalt koondunuks. Selliselt koondus algoritm 192 iteratsiooniga 62.5% kordadest. Tõenäoliselt ei koondunud 217 px, 299 px ja 432 px pindalaga pildid, sest leidusid tugevad ja valed lokaalsed miinimumid, kuhu koondus enamus optimeeringuid. Selle vältimiseks on vaja jooksutada algoritmi rohkem arv kordi või tugevamalt piirata muutujaruumi.

Joonisel 10 on näha Open3D ja Blenderiga simuleeritud pilte kaugusel 1359 km. Pildid tunduvad sarnased, kuid on siiski erinevad ning sellest tulenevad ka kaamera nurga viga 13.47° ja asukoha viga 14.10%. Kaamera õige poosi lähedal on tugevad lokaalsed miinimumid, kuhu algoritm valesti koonduda võib. Sellisesse lokaalsesse miinimumi on koondunud joonisel 10 kujutuatud tulemus. Tõenäoliselt on taolisi lokaalseid miinimume rohkem, millest tulenevalt ei ole algoritmil võimalik edukalt õigesse miinimumi koonduda. Algoritmi täpsuse piiriks võivad olla saadud nurga ja asukoha viga, vastavalt 13.4° ja 14.10%. Selle tõestamiseks oleks vaja teha edasisi uurimusi.



Joonis 10. Blenderi ja Open3D poolt simuleeritud kaadrite võrdlus, kus kaugus tuumast on 1359 km. Vasakul on näha Blenderiga simuleeritud OPIC pilt, mida üritati jäljendada Open3D programmis. Jäljendatud pilt on näha paremal.

Veafunktsiooni hinnang

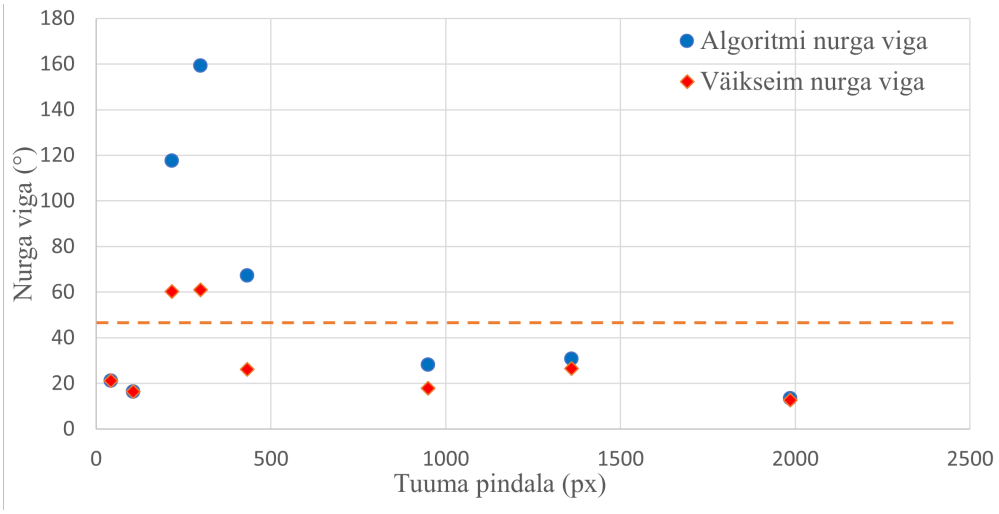
Veafunktsiooni hindamiseks võrreldakse veafunktsiooni poolt valitud tulemuse vigu ning reaalselt kõige väiksemate vigadega tulemust. Tulemused on leitavad tabelis 4 ja joonisel 11. Tabelis on tähistatud algoritmi poolt eristatud tulemuse viga tähega A ning reaalselt väikseimat viga tähega R. Nagu näha, valis funktsioon reaalselt väiksema veaga poosi 25% kordadest. 432 px tuuma pindalaga pildi korral ei leidnud veafunktsioon üles ühte koondumistingimustele vastavat tulemust, mis oleks tõstnud koondumisprotsendi 62.5% pealt 75% peale. See tuleb välja jooniselt 11, sest 432 px juures on algoritmi nurga viga ja väikseim nurga viga erineval pool koondumisjoont.

Kuna vastavus veafunktsiooni valitud ja reaalselt väikseimale nurgale esines ainult esimese ja teise pildi juures, siis 25% kordadest valis kõikide iteratsioonide hulgast veafunktsioon õige tulemuse. Ülejäänud 75% puhul aga mitte. See on halb, sest reaalses olukorras, kus ei ole õigeid kaamera poose teada, läheksid õiged tulemused kaotsi. Veafunktsioon leidis koondumisparameetritele vastava tulemuse 62.5% juhtudest, mistõttu ei ole see täielikult ebaõnnestunud, kuid siiski ei ole tulemus optimaalne. Seda funkt-

siooni kasutab optimeerija parima poosi väljaselgitamiseks, mis tähendab, et täpsema veafunktsiooni puhul suureneks tõenäoliselt ka koondumiste arv.

Tabel 4. MSE järgi valitud ning reaalselt parimad tulemused. Tabelis on tähistatud algoritmi poolt valitud nurkade vead tähega A ning reaalselt väikseima veaga tulemust tähega R.

Nr	Pindala (px)	A nurk(°)	A kaugus (%)	R nurk (°)	R kaugus (%)	Vastavus
1	42	21.21	29.32	21.21	29.32	Jah
2	106	16.39	21.82	16.39	21.82	Jah
3	217	117.55	158.26	60.35	98.25	Ei
4	299	159.22	163.40	61.08	91.06	Ei
5	432	67.28	111.90	26.16	36.68	Ei
6	949	28.12	42.45	17.90	25.37	Ei
7	1360	30.78	47.62	26.50	39.77	Ei
8	1986	13.47	14.10	12.66	15.76	Ei

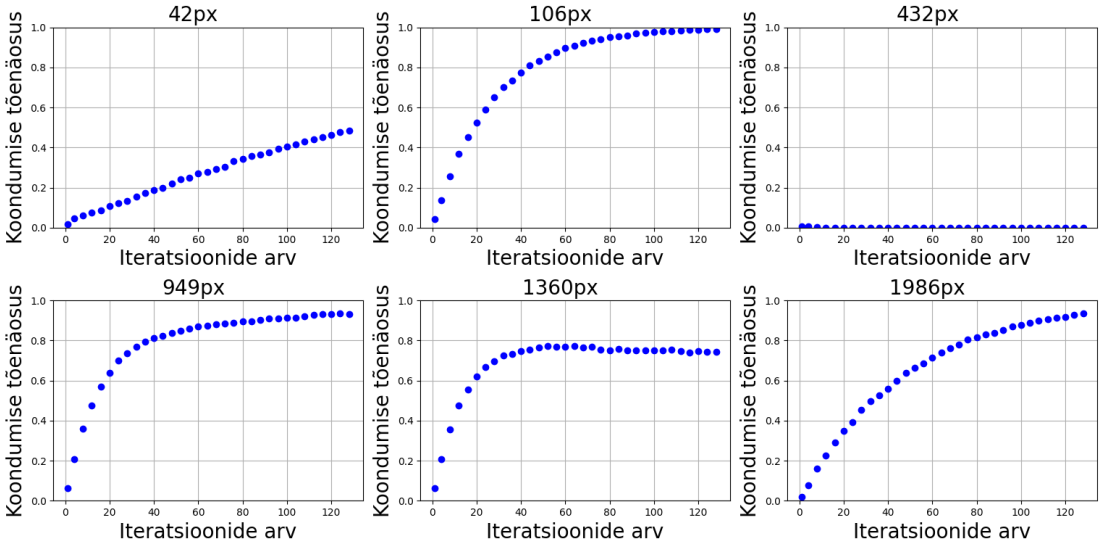


Joonis 11. Algoritmi ja käsitsi valitud parima nurga võrdlus. Algoritm valis optimeeringute tulemuste seast väikseima ruutkeskmise veaga tulemuse, mis 75% kordadest ei ühti reaalselt parima tulemusega. Ühel korral, 432 px juures, on algoritmi antud hinnang koonduvusele vale. Koondumisparameetritele vastav tulemus oli olemas, kuid seda ei eristatud teiste seast.

Iteratsioonide arvu efektiivsus

Optimaalse iteratsioonide arvu hindamiseks võetakse arvutatud 192 simulatsioonist juhuslikult erinevate suurustega alamhulgad. Koondumise tõenäosuse sõltuvus iteratsioonide arvust on näha joonisel 12. Jooniselt on ära jäetud 217 px ja 299 px tulemused, kuna nende piltide juures algoritm ei koondunud ning graafikud on sarnased 432 px juhule, mis samuti ei koondunud.

Jooniselt on näha, et koondunud piltide puhul läheneb koondumise tõenäosus aeglustuva kasvuga ühele. Esimese ehk 42 px pindalaga juhul sellist kasvu ei esine ja graafik on lineaarne. Tõenäoliselt on aeglane kasv tingitud väikesest koondumiste arvust, mis tähendab, et koondumine praegustel katse tingimustel oli ebatõenäoline. See on oodatav, kuna vastaval pildil on infot kõige vähem, mistõttu oleks vastavate tingimustega koondumine kõige ebatõenäolisem. Kui itereerida optimeerijat rohkem arv kordi, siis võib eeldada, et ka 42 px pindalaga pildi graafik oleks teistega sarnasem.



Joonis 12. Koondumise tõenäosuse sõltuvus iteratsioonide arvust. Kõik andmepunktid on keskmistatud üle 10000 alamhulga. Iteratsioonide arvu suurenedes läheneb koondumise tõenäosus ühele, välja arvatud 432 px korral, kus algoritm ei suutnud leida ühtegi head tulemust.

4.2 Lokaalse optimeerija tulemused

Lokaalse optimeerija algoritmi jooksutati ühe pildi peal 160 korda ning kasutada oli viis pilti, millel olid tuuma pindalad vahemikus 43-2390 px. Kokku kulus 800 optimeerija jooksutamiseks 10 tundi sama töömasina peal, millega jooksutati ka globaalset optimeerijat. Optimeerimistulemused koos algpakkumiste vigadega on leitavad tabelites 5 ja 6 ning joonisel 13. Tulemuste tegelik täpsus on 7 kohta peale koma, mis tuleneb 32-bitilise ujukomaarvu täpsusest, kuid loetavuse huvides on tulemused esitatud täpsusega 2 kohta peale koma.

Tabel 5. Lokaalse optimeerija nurkade vigade võrdlus.

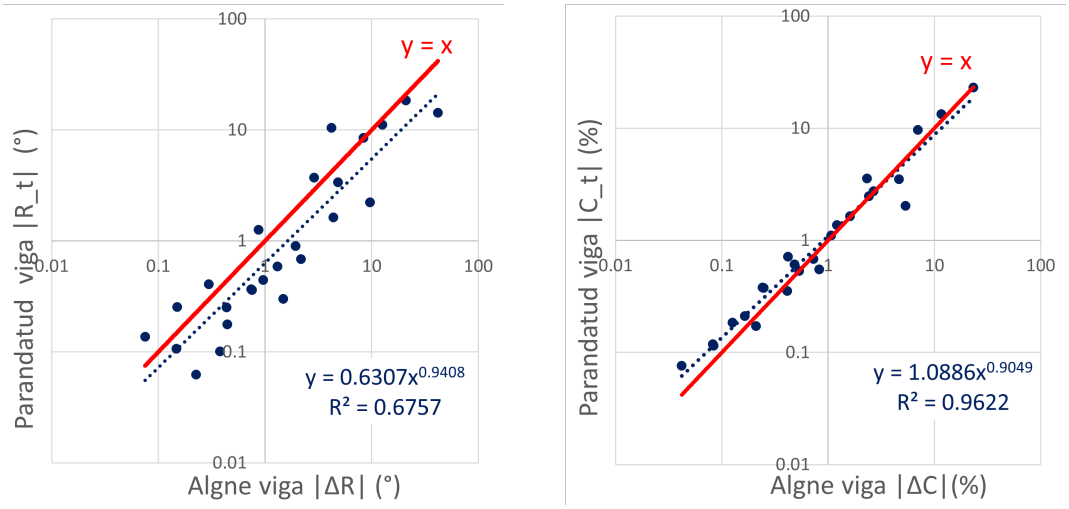
Pindala (px)	Keskmine algviga $(^\circ)$	1. optimeerija viga $(^\circ)$	2. optimeerija viga $(^\circ)$
43	17.58	12.57	12.57
187	4.04	2.13	2.13
413	1.83	0.86	0.88
1217	0.62	0.40	0.27
2390	0.32	0.26	0.18

Tabel 6. Lokaalse optimeerija asukohtade vigade võrdlus.

Pindala (px)	Algviga $(\%)$	1. optimeerija viga $(\%)$	2. optimeerija viga $(\%)$
43	9.77	10.69	10.69
187	2.25	1.61	1.62
413	1.02	1.10	1.10
1217	0.35	0.20	0.32
2390	0.18	0.15	0.25

Joonistel 13a ja 13b on näha vastavalt nurga ja positsiooni vigade sõltuvus algpakkumise veast log-log teljestikus. Joonisel on algviga $y = x$ tähistatud pideva joonega ning algoritmi poolt leitud tulemused punktidega. Punktiiriga on tähistatud tulemustele sobitatud regressioonijoon, mille valem on leitav joonise all paremal. Kui algoritmi poolt leitav viga on väiksem esialgsest pakkumisest, siis on tulemus allpool algviga tähistavat joont. Joonisel 13a on näha, et nurkade hinnangu parandamisel oli lokaalne optimeerija edukas, sest enamus leitud punktidest on algpakkumisest väiksema veaga. Joonisel 13b on näha, et asukoha täpsustamisel algoritm edukas ei olnud, sest enamasti on arvutatud

vead suuremad algpakkumiste vigadest. Viga tähistavad punktid jäävad enamasti joonisel sirgest $y = x$ üles poole.



(a) (b)

Joonis 13. Lokaalsete optimeeringute vigade tulemused log-log teljestikes. Joonisel 13a on näha parandatud nurgavea sõltuvus algvea nurgast. Joonisel 13b on näha parandatud asukoha vea sõltuvus algsest veast. Pideva joonega on mõlemal joonisel tähistatud $y = x$ sirge, mis tähistab esialgset viga. Kui tulemused on allpool seda sirget, siis on optimeerija esialgset veahinnangut edukalt täpsustanud. Kui tulemused on üleval pool pidevat joont, siis on algoritmi veahinnang suurem alghinnangust ning optimeerija ebaõnnestus. Jooniselt on näha, et nurga vea vähendamisel oli loodud algoritm edukas, sest enamus punkte on allpool $y = x$ sirget. Asukoha vea määramisel ei olnud algoritm edukas, sest enamasti jäävad punktid sirgest üles poole.

Optimeeringutest järeldub, et algpakkumise nurga vea $|\Delta R|$ korral on võimalik lokaalse optimeerijaga seda viga vähendada veani $|\Delta R_t|$. Seda seost kirjeldab valem 20, mis on saadud optimeerimistulemustele regressioonisirge arvutamisega. Vaata lähemalt jooniselt 13a.

$$|\Delta R_t| = 0.6307 \cdot |\Delta R|^{0.9408} \tag{20}$$

Lokaalse optimeerija täpsuse piiri iseloomustab koht $x = 0.00042$, kus sirge $y = x$ ja joon $y = 0.6307x^{0.9408}$ lõikuvad. See tähendab, et juhul kui pöörde algpakkumise täpsus on väiksem kui $0.00042°$, siis lokaalne optimeerija ei ole võimaline seda pakkumist täpsemaks tegema. Asukoha täpsustamisel ei olnud algoritm edukas, mistõttu ei ole

asukoha hinnangu parandamiseks mõistlik lokaalset optimeeringut teha.

4.3 Tulemuste analüüs

Käesoleva töö tulemusena leiti, et antud ülesande parameetritega loodud globaalse optimeerija 192 iteratsiooniga on võimalik määrata kaamera poos kõige väiksema nurga veaga 13.47° ning asukoha veaga 14.10%. Kõige ebatäpsem tulemus oli nurga veaga 159.22° ja asukoha veaga 163.40%. Vea suurus on tingitud koondumise tõttu valesse miinimumi.

Tulemused ei koondunud täpsemalt, kuna otsitava kaamera poosi juures on tõenäoliselt palju tugevaid lokaalseid miinimume, kuhu optimeerija koondub enne globaalsesse miinimumi jõudmist. Globaalse optimeerija tulemuste vead on liiga suured, et neist ülesande kontekstis kasu oleks, sest OPIC poos on varasemalt palju täpsemalt teada. Tööga tõestati, et antud meetodiga ei ole võimalik asukohta ega nurka globaalse optimeerimisega piisavalt täpselt määrata.

Samuti leiti, et loodud lokaalne optimeerija on edukas poosi nurga hinnangu täpsustamiseks. Arvutati ligikaudne seos täpsustatud nurga vea ning algse hinnangu nurga vea vahel, mida on kirjeldatud valemiga 20. Loodud optimeerija hinnanguliseks täpsuse piiriks arvutati 0.00042°, mis tähendab, et kui algse nurga hinnang on sellest väärtusest suurem, on algoritm võimeline seda täpsustama. Poosi asukoha hinnangu täpsustamisel algoritm ebaõnnestus.

Nii lokaalse kui ka globaalse optimeerija täpsust on tõenäoliselt võimalik tõsta iteratsioonide arvu suurendades, kuid sellisel juhul läheks arvutusaeg ebamõistlikult suureks.

4.4 Tulevik

Loodud optimeerijaid oleks võimalik edasi arendada ja nende täpsust tõsta, kui arvestada optimeeringute algpakkumistes eelnevate pooside tulemustega. Pildid on tehtud ligikaudu sirgjooneliselt liikudes, mis tähendab, et kahe tuumale lähemal asetseva poosiga, kus tulemus on kõige täpsem, saaks ära määrata sirge, mille peal peaksid ka kaugemad poosid asuma. Samuti oleks võimalik teha optimeerija parameetritele metaanalüüs eesmärgiga muutujaruumi rohkem piirata. Proovida võiks veel erinevaid veafunktsioone.

Kokkuvõte

Töö eesmärgiks oli välja pakkuda ning katsetada meetodeid, millega saaks hinnata komeedivaatluskaamera OPIC asukohta komeedi tuuma järgi. Ülesande jaoks ei sobi tavalised võrdlusmeetodid, kuna OPIC kaameraga tehtud piltidel on komeedi tuuma läbimõõt väga väike, vahemikus 10-30 pikslit. Seetõttu ei ole pilditunnuste eristamise ja nende sobitamisega võimalik kaamera poosi määrata.

Töö käigus pakuti kaamera poosi leidmiseks välja lahendus, mis kasutab sisendina komeedivaatluskaamera OPIC poolt tehtud pilti ja CoCa poolt tehtud pilte. Viimastest loodi SfM tarkvaraahelat kasutades 3D mudel, mille vastu testiti erinevaid kaamera poose. Igast poosist simuleeriti OPIC parameetritega pilt ning seda võrreldi reaalse OPIC pildiga.

Poosi leidmiseks loodi globaalne ja lokaalne optimeerija, mis iteratiivselt muutujaruumi parameetreid läbi proovisid. Globaalse optimeerija puhul ei olnud õige poosi kohta mingit infot ning optimeerija eesmärgiks oli määrata OPIC poos tundmatus olukorras. Lokaalse optimeerija puhul oli OPIC hinnanguline poos varasemalt teada ning optimeerijat kasutati olemasoleva hinnangu täpsustamiseks.

Erinevate pooside läbi proovimiseks kasutati Powelli optimeerimisalgoritmi, sest see on efektiivne ning ei vaja lahendamiseks funktsiooni tuletiste olemasolu. Küll aga on Powelli optimeerimisalgoritm tundlik algpakkumiste suhtes. Selleks et suurendada õigesse poosi koondumise tõenäosust, jooksutati korraga mitmeid optimeerijaid erinevate algpakkumistega. Vastuseks valiti kõige madalama veafunktsiooniga tulemus. Saadud tulemusi võrreldi Blenderist saadud reaalsete kaamera poosidega.

Globaalse optimeerija testimiseks kasutati kaheksat pilti, millel olid komeedi tuumade pindalad vahemikus 42-1986 px. Ühe pildi poosi määramiseks jooksutati algoritmi 192 korda. Kõige täpsema poosi nurga veaks saadi 13.47° ja kauguse veaks 14.10%. Kõige ebatäpsema poosi nurga viga oli 159.22° ja kauguse viga 163.40%. Suure vea põhjuseks on koondumine valesse miinimumi. Loodud algoritmi abil koondusid väiksema kui 45° nurga ja 50% kauguse veaga 62.5% poosidest. Saadud tulemused on liiga suurte vigadega, et olla kasulikud OPIC asukoha ja pöörde määramisel.

Lokaalse optimeerija testimiseks kasutati viit pilti komeedi tuuma pindaladega 43-2390 px. Iga pildi peal jooksutati 160 optimeerijat ning vastuseks võeti väikseima veafunktsiooniga tulemus. Leiti, et algoritm on poosi nurga täpsustamisel edukas ning

hinnati algoritmi täpsuse piiriks 0.00042°. Poosi asukoha leidmisel ei olnud loodud algoritm edukas, sest leitud asukohtade vead olid enamasti suuremad alghinnangu veast.

Algoritmi täpsuse tõstmiseks saaks teha välisparameetrite metaanalüüsi, et optimeeringute parameetreid täpsemalt määrata. Katsetada võiks ka erinevaid veafunktsioone. Samuti saaks algoritmi edasi arendada, kui võtta pildi poosi määramisel arvesse ka varasemate piltide pooside tulemusi.

Kasutatud kirjandus

[1] C.Šnodgrass and G. H. Jones, "The european space agency's comet interceptor lies in wait," *Nature communications*, vol. 10, no. 1, pp. 1–4, 2019.

[2] U. Nadeem, M. Bennamoun, R. Togneri, and F.Šohel, "Unconstrained matching of 2d and 3d descriptors for 6-dof pose estimation," *arXiv preprint arXiv:2005.14502*, 2020.

[3] L. Liu, D. Campbell, H. Li, D.Žhou, X.Šong, and R. Yang, "Learning 2d-3d correspondences to solve the blind perspective-n-point problem," *arXiv preprint arXiv:2003.06752*, 2020.

[4] M. Pajusalu, J. Kivastik, I. Iakubivskyi, and A.Šlavinskis, "Developing autonomous image capturing systems for maximum science yield for high fly-by velocity small solar system body exploration," in *Proceedings of the International Astronautical Congress, IAC*, vol. 2020, International Astronautical Federation, IAF, 2020.

[5] P. R. Weissman, "The oort cloud," *Nature*, vol. 344, no. 6269, pp. 825–830, 1990.

[6] E. Öpik, "Note on stellar perturbations of nearly parabolic orbits," in *Proceedings of the American Academy of Arts and Sciences*, vol. 67, pp. 169–183, JSTOR, 1932.

[7] J. H. Oort, "The structure of the cloud of comets surrounding the solar system and a hypothesis concerning its origin," *Bulletin of the Astronomical Institutes of the Netherlands*, vol. 11, pp. 91–110, 1950.

[8] J. P. Sánchez, D. Morante, P. Hermosin, D. Ranuschio, A. Estalella, D. Viera, S. Centuori, G. Jones, C.Šnodgrass, A. C. Levasseur-Regourd, *et al.*, "Esa f-class comet interceptor: Trajectory design to intercept a yet-to-be-discovered comet," *Acta Astronautica*, vol. 188, pp. 265–277, 2021.

[9] N. Haslebacher, S.-B. Gerig, N. Thomas, R. Marschall, V.Žakharov, and C. Tubiana, "A numerical model of dust particle impacts during a cometary encounter with application to esa's comet interceptor mission," *Acta Astronautica*, vol. 195, pp. 243–250, 2022.

[10] S. Lipschutz, D.Špellman, and M. R. Spiegel, *Vector analysis and an introduction to tensor analysis*. McGraw-Hill's ebook library. Student study aids. Schaum's course outlines.; Schaum's outline series, McGraw Hill Professional, 2nd ed ed., 2011.

[11] S. Ullman, "The interpretation of structure from motion," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 203, no. 1153, pp. 405–426, 1979.

[12] P. Moulon, P. Monasse, and R. Marlet, "Adaptive structure from motion with a contrario model estimation," in *Asian conference on computer vision*, pp. 257–270, Springer, 2012.

[13] K. E. Ozden, K. Cornelis, L. Van Eycken, and L. Van Gool, "Reconstructing 3d trajectories of independently moving objects using generic constraints," *Computer Vision and Image Understanding*, vol. 96, no. 3, pp. 453–471, 2004.

[14] M. Feng, S. Hu, M. H. Ang, and G. H. Lee, "2d3d-matchnet: Learning to match keypoints across 2d image and 3d point cloud," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 4790–4796, IEEE, 2019.

[15] L. Liu, H. Li, and Y. Dai, "Efficient global 2d-3d matching for camera localization in a large-scale 3d map," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2372–2381, 2017.

[16] S. Jayawardena, M. Hutter, and N. Brewer, "Featureless 2d–3d pose estimation by minimising an illumination-invariant loss," in *2010 25th International Conference of Image and Vision Computing New Zealand*, pp. 1–8, IEEE, 2010.

[17] H. Yoshitaka, K. Hirohiko, O. Akihisa, and Y.Šhin'ichi, "Mobile robot localization and mapping by scan matching using laser reflection intensity of the sokuiki sensor," in *IECON 2006-32nd Annual Conference on IEEE Industrial Electronics*, pp. 3018–3023, IEEE, 2006.

[18] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The computer journal*, vol. 7, no. 2, pp. 155–162, 1964.

[19] B.Õ. Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022.

[20] P. Moulon, P. Monasse, R. Marlet, and Others, "Openmvg." `https://github.com/openMVG/openMVG`.

[21] D. Cernea and Others, "Openmvs." `https://github.com/cdcseacave/openMVS`.

[22] M. Pajusalu and A.Šlavinskis, "Characterization of asteroids using nanospacecraft flybys and simultaneous localization and mapping," in *2019 IEEE Aerospace Conference*, pp. 1–9, IEEE, 2019.

[23] P.Õ. Community, *Python 3.10*. Python Software Foundation, 2022.

[24] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

Lisad

I. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Uku Kert Paidra**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Komeedivaatluskaamera OPIC asukoha määramine sellega pildistatud fotode põhjal,

   mille juhendaja on Mihkel Pajusalu, PhD,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Uku Kert Paidra
29.05.2022