Jere Pakkanen

Suurikokoisen volumetrisen datan visualisointi käyttäen poikkileikkauksia

Tietotekniikan pro gradu -tutkielma

19. marraskuuta 2023

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Jere Pakkanen

Yhteystiedot: jere.pakkanen@hotmail.com

Ohjaaja: Paavo Nieminen ja Jukka Kuva

Työn nimi: Suurikokoisen volumetrisen datan visualisointi käyttäen poikkileikkauksia

Title in English: Large sized volumetric data visualization using cross-sections

Työ: Pro gradu -tutkielma

Opintosuunta: Ohjelmisto- ja tietoliikennetekniikka, filosofian maisteri

Sivumäärä: 99+2

Tiivistelmä: Tässä pro gradu -tutkielmassa kehitetään ohjelmistomoduuli, jonka avulla voidaan visualisoida suurikokoisia volumetrisen datan joukkoja poikkileikkausten avulla. Erona aiempiin samankaltaisiin toteutuksiin on se, että koko volumetrisen datan joukkoa ei lueta tietokoneen muistiin kerralla. Volumetrisesta datasta luetaan dynaamisesti massamuistista vain poikkileikkauksen luomiseen käytettäviä arvoja. Tämä mahdollistaa suurikokoisten datajoukkojen käsittelyn tietokoneella, jossa ei ole paljon keskusmuistia. Tutkielmassa osoitetaan myös ratkaisun toimivuus integroimalla kehitetty ohjelmistomoduuli volumetrista dataa käsittelevään GeoLab-ohjelmistoon.

Avainsanat: Volumetrinen data, Suunnittelutiede, Viipalointi, Visualisointi, Interpolointi, WPF

Abstract: In this master's thesis a software module, which can visualize large sized volumetric data using cross-section, is developed. Compared to earlier similar implementations, the developed module doesn't load the whole volumetric data set into the computer's memory. Only values that are used in making the cross-section, are read from the mass storage. This allows handling large sized data sets with a computer, that doesn't have a lot of RAM. This thesis also validates the usability of the solution by integrating the developed module into GeoLab program, that is used to handle volumetric data.

Keywords: Volumetric data, Design science, Slicing, Visualization, Interpolation, WPF

Kuviot

Taulukot

Sisällys

1 Johdanto

Volumetrisen datan visualisoinnille on käyttötarkoituksia useissa eri tieteen saroissa, kuten geologiassa ja lääketieteessä. Volumetrisen datan koon kasvaessa myös tarve visualisointitekniikkojen optimoinnille on kasvanut. Suurin osa aihetta käsittelevästä tutkimuksesta on keskittynyt tilavuushahmonnustekniikoihin (eng. volume rendering). Tilavuushahmonnus tarjoaa käyttäjälle tavan tarkastella koko volumetristä dataa yhdestä näkymästä. Tässä tutkielmassa volumetrisen datan visualisointia lähestytään viipaloinnilla (eng. slicing).

Tämän tutkimuksen tarkoituksena on selvittää, miten suurikokoista volumetrista dataa voidaan esittää viipaloinnin avulla perustietokoneella. Etuna tässä menetelmässä on se, että verrattuna tilavuushahmonnustekniikoihin, kaikkea volumetristä dataa ei tarvitse käydä läpi visualisoinnin tuottamiseksi, vaan volumetristä dataa luetaan vain käyttäjän määrittämältä alueelta. Tällöin käyttäjän syötteeseen on helpompi reagoida reaaliajassa. Tämä myös antaa käyttäjälle mahdollisuuden tarkastella häntä kiinnostavia alueita datasta. Tämä tulee sen kustannuksella, että kaikkea dataa ei voida visuaalisesti esittää kerralla yhdestä näkymästä. Koska tutkielman lähestymistavassa volumetrisen datan visualisoinnissa käyttäjän oma rooli on isompi antamalla hänelle täyden kontrollin kiinnostavan alueen määrittämiseksi, onkin tärkeää että moduulin käyttämisestä tehdään mahdollisimman intuitiivista ja interaktiivista.

Tutkimus on suunnittelutieteellinen tutkimus. Kehitettävä artefakti tutkielmassa on ohjelmistomoduuli. Toteutettavan ohjelmistomoduulin avulla käyttäjä pystyy tarkastelemaan suurikokoista volumetristä dataa vapaavalintaisesta näkökulmasta ja syvyydestä poikkileikkauksien avulla. Ohjelmistomoduuli on myös tarkoitus integroida ulkoiseen GeoLab-ohjelmistoon tutkimuksen aikana. Tällä varmistetaan, että moduuli toimii myös oikeassa käyttöympäristössä. GeoLab-ohjelmistosta kerrotaan lisää alaluvussa 2.5. Moduulin kehitystä ja sen integrointia ulkoiseen ohjelmistoon toteutetaan useassa iteraatiossa. Ohjelmistomoduulin toimintaa ja sen integrointia GeoLabiin arvioidaan kehitysiteraatioiden välissä. Tutkimus suoritetaan yhteistyössä GTK:n (Geologian tutkimuskeskus) kanssa. Tutkimuksessa toteutetun ohjelmistomoduulin lähdekoodi on julkisesti saatavilla GeoLabin versiohallinnassa[1].

1. https://gitlab.com/gtkwp4/geolab/

Tutkielman luvussa 2 käydään läpi tutkimuksen aiheelle olennaisia käsitteitä. Luku 3 käsittelee tutkimuksen tutkimusmenetelmää. Luvussa käydään läpi, miksi tutkimuksessa käytetään suunnittelutieteellistä tutkimusmenetelmää, mikä tutkimusmenetelmän ideana on sekä miten tutkimusta toteutetaan. Luvussa 4 käydään läpi miten tutkielmassa kehitettävää ohjelmistomoduulia kehitetään sekä integroidaan GeoLab-ohjelmistoon. Luvussa 5 käsitellään kuinka tutkielmassa kehitettävää ohjelmistomoduulia sekä sen integrointia GeoLab-ohjelmistoon arvioidaan. Tämän arvioinnin pohjalta ohjelmistomoduulin sekä sen integroinnin kehitystä ohjataan eteenpäin. Luvussa 6 käydään läpi, minkälaisia menetelmiä hyödyntämällä tutkimuksen tavoite saatiin toteutettua. Tämän lisäksi luku tiivistää, miten kehitetyn ohjelmiston kehitys ja arviointi toteutettiin sekä mitä aiheita mahdolliset jatkotutkimukset voisivat käsitellä. Lopuksi luku sisältää pohdintaa siitä miten tutkimuksessa käytettyjä menetelmiä voitaisiin käyttää muualla. Luku 7 käy läpi lyhyesti tutkimuksen olennaisimmat asiat, eli tutkimuksen tavoitteet, tutkimuksen tulokset, mahdolliset poikkeamat tutkimussuunnitelmasta sekä mahdolliset jatkotutkimusaiheet, jotka tulivat tutkimuksen aikana esille.

2 Taustaa

Tämä luku sisältää taustatietoa tutkimuksen aihepiirin keskeisistä käsitteistä. Näitä käsitteitä ovat volumetrinen data, sen esitysmuodot ja visualisointi, interpolointi sekä GeoLab-ohjelmisto, johon tutkimuksessa kehitettävä ohjelmistomoduuli integroidaan.

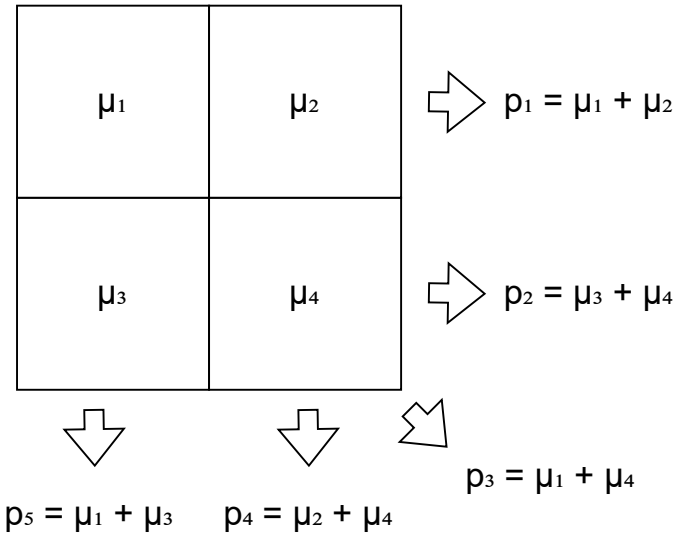
2.1 Volumetrinen data

Kaufman (2000) määrittelee volumetrisen datan olevan yleensä joukko S mittaustuloksia (x, y, z, v), jotka kuvaavat jonkin kohteen ominaisuuden arvoa v kolmiulotteisessa pisteessä (x, y, z). Kaufman (2000) antaa esimerkkinä arvon v kuvaamasta ominaisuudesta tiedon siitä sijaitseeko avaruuden pisteessä (x, y, z) volumetrisen datan käsittelemä kohde. Jos kohde sijaitsee tässä pisteessä, silloin $v = 1$. Jos taas pisteessä ei ole mitään, niin $v = 0$. Edellä mainitussa esimerkissä volumetrinen data on binääridataa. Tällöin v mahdollisia arvoja ovat $v \in \{0, 1\}$. Riippuen ominaisuudesta mitä volumetrinen data mittaa, v:n arvojoukko voi vaihdella. Esimerkiksi tiheyden tapauksessa kaikki v:n mahdolliset arvot sijoittuvat välille $[0, \infty[$.

Kaufman (2000) kertoo, että volumetrinen data voi myös olla moniarvoista, jolloin yhdessä pisteessä (x, y, z) on n kappaletta arvoja $(v_1, v_2, \ldots, v_n)$. Nämä arvot voivat esittää esimerkiksi volumetrisen datan käsittelemän kappaleen lämpötilaa, väriä tai tiheyttä pisteessä (x, y, z). Koska usean eri arvon esittäminen samassa avaruuden pisteessä (x, y, z) olisi visuaalisesti haastavaa esittää selvästi yhdessä kuvassa, eri arvoja esitetään yleensä erillisissä kuvissa. Esimerkiksi kolmikanavaisissa RGB-kuvissa, jokainen kanava voi käsitellä eri mitattua ominaisuuden arvoa, jolloin näytettävää värikanavaa vaihtamalla voidaan sulavasti esittää erinäisiä mitattuja ominaisuuksia volumetrisen datan käsittelemästä kohteesta. Tämä tutkielma keskittyy käsittelemään vain yksikanavaista kolmiulotteista volumetristä dataa.

Yksi yleinen tapa luoda yksikanavaista volumetrista dataa on tietokonetomografia. Iinuma (1983) määrittelee tietokonetomografian perusideana olevan röntgensäteen ohjaaminen tarkasti kuvattavan kohteen läpi yhdestä kohtaa. Röntgensäde vastaanotetaan kuvattavan kohteen läpäisyn jälkeen elektronisella vastaanottimella, jolla saadaan mitattua läpäisevän säteilyn intensiteettiprofiili. Tämä mittaus toistetaan kunnes kohteesta on saatu mitattua läpäise-

vän säteilyn intensiteettiprofiili kauttaaltaan yhdestä tarkastelukulmasta. Tämän jälkeen joko säteilylähdettä ja vastaanotinta tai kuvattavaa kohdetta siirretään, jotta säteilyn intensiteettiprofiili voidaan mitata toisesta tarkastelukulmasta. Kun kuvattavasta kohteesta säteilyn intensiteettiprofiili on mitattu useasta kulmasta, voidaan kohteesta muodostaa kolmiulotteinen rekonstruktion. Hsieh (2003) kertoo rekonstruktioon olevan useita menetelmiä nykypäivänä. Hän antaa kuvassa 1 esitetyn yksinkertaisen esimerkin siitä miten rekonstruktio voidaan tehdä. Hänen esimerkissään oletetaan olevan neljästä homogeenisestä neliöstä muodostuva kuvattava kohde. Tätä kohdetta kuvataan viidestä eri suunnasta, jolloin saadaan viisi kokonaisvaimeneman mittaustulosta $p_1, \ldots, p_5$. Näiden viiden mittaustuloksen avulla voidaan ratkaista neljän neliön vaimenemat $\mu_1, \ldots, \mu_4$. Koska neliöt ovat keskenään yhtä suuria, niiden aiheuttama vaimenema on suoraan yhteydessä neliöiden tiheyteen.



Kuvio 1: Yksinkertaistettu esimerkki kuinka eri kulmista otetut kuvat muodostavat matriisin (mukaillen Hsieh 2003).

Kuvio 1 havainnollistaa, kuinka saadaan muodostettua yhtälöt, joiden avulla voidaan ratkaista arvot $\mu_1, \ldots, \mu_4$. Yhtälöistä muodostuva yhtälöryhmä voidaan esittää esimerkiksi matriisin muodossa seuraavasti:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix}$$

Jos yhtälöryhmistä muodostetaan matriisi, arvot $\mu_1, \ldots, \mu_4$ voidaan ratkaista erilaisilla matriiseihin soveltuvilla algoritmeilla. Hsieh (2003) antaa esimerkkinä, että ensimmäinen tietokonetomografialaitteisto käytti vuonna 1967 suoraa matriisin kääntö (eng. direct matrix inversion) algoritmia ratkaisemaan yksittäisten kohtien vaimenemia $\mu_1, \ldots, \mu_n$.

Tietokonetomografialla on useita eri käyttötarkoituksia eri aloilla. Lääketieteessä tietokonetomografiaa voidaan käyttää kuvaamaan ihmisen sisäisiä elimiä, proteeseja ja luustorakennetta. Yang ym. (2018) käyttävät esimerkiksi tutkimuksessa elektronista tomografiaa lonkkaproteesileikkauksen jälkeisen korjausleikkauksen reaaliaikaiseen valvomiseen. Geologiassa tietokonetomografiaa voidaan käyttää kivinäytteen sisäisen rakenteen tutkimiseen. Sayab ym. (2021) esittävät kuinka kivinäytteen rakennetta analysoimalla voidaan esimerkiksi määrittää näytteen alkuperäalueen tektonista historiaa.

Kaufman (2000) kertoo, että vaikka teoriassa mittaustulokset voidaan ottaa satunnaisista kuvattavan kohteen kohdista, joukko S on yleensä isotrooppinen. Tämä tarkoittaa, että joukon S sisältämät mittaustulokset ovat otettu tasaisin välein kolmen eri pääakselin suuntaisesti. Jos yksittäisen pääakselin mittaustulokset ovat tasaisin välein, mutta eri pääakselien välillä mittaustuloksien välit ovat erisuuruisia keskenään, on joukko S anistrooppinen. Tässä tutkielmassa keskitytään tutkimaan vain isotrooppisia joukkoja S. Kaufman (2000) kertoo, että joukkoa S voi määritellä mittaustuloksina säännöllisessä kolmiulotteisessa hilassa. Tämän takia yleensä volumetrisen datan tallennusmuotona käytetään kolmiulotteista taulukkoa. Tällöin mittaustulosten arvojen sijainti taulukossa määrittää niiden arvojen sijainnin kolmiulotteisessa hilassa.

2.2 Interpolointi

Kaufman (2000) selittää joukon S määrittävän ominaisuuksien arvoja vain diskreeteissä avaruuden kohdissa. Jotta volumetrisen datan arvoja voidaan esittää myös diskreettien avaruuden kohtien ulkopuolelta, on käytettävä jotakin menetelmää, jolla voidaan luoda arvoja diskreettien kohtien välistä. Kaufman (2000) esittää tähän ratkaisuksi funktion $f(x,y,z)$, joka voidaan määrittää kattamaan koko volumetrisen datan avaruus $\mathbb{R}^3$. Tällöin voidaan kuvailla minkä tahansa jatkuvan avaruuden pisteen arvoa. Merkitään joukon S mittaustuloksen arvoa pisteessä (x,y,z) merkinnällä $S(x,y,z)$. Tällöin funktio $f(x,y,z) = S(x,y,z)$, jos (x,y,z) on jokin hilan mittaustuloksen diskreetti piste. Muussa tapauksessa $f(x,y,z)$ approksimoi mittaustuloksen arvoa pisteessä (x,y,z) käyttämällä jotakin funktiota joukkoon S, joka approksimoi pisteen arvoa. Tätä menetelmää kutsutaan yleisellä tasolla interpoloinniksi.

Kirjallisuudesta löytyy myös erilaisia määritelmiä interpoloinnille, jotka eroavat hieman toisistaan. Esimerkiksi Thévenaz, Blu ja Unser (2000) määrittelevät interpoloinnin olevan malliin perustuva menetelmä, jolla pyritään saamaan takaisin jatkuvan datan arvoja mielivaltaisessa pisteessä käyttäen hyväksi tiedossa olevien diskreettien pisteiden arvoja sekä mielivaltaisen pisteen etäisyyttä niihin. Kaufman (2000) esittämä määritelmä eroaa tästä hieman, sillä hänen määritelmänsä ei suoraan vaadi käyttämään interpoloitavan pisteen etäisyyttä diskreetteihin pisteisiin arvon interpoloinnissa. Käytettävästä määritelmästä huolimatta, useat yleisesti käytetyt interpolointimenetelmät käyttävät hyväksi interpoloitavan pisteen etäisyyttä lähimpien mittaustulosten pisteiden arvoihin interpoloinnissa.

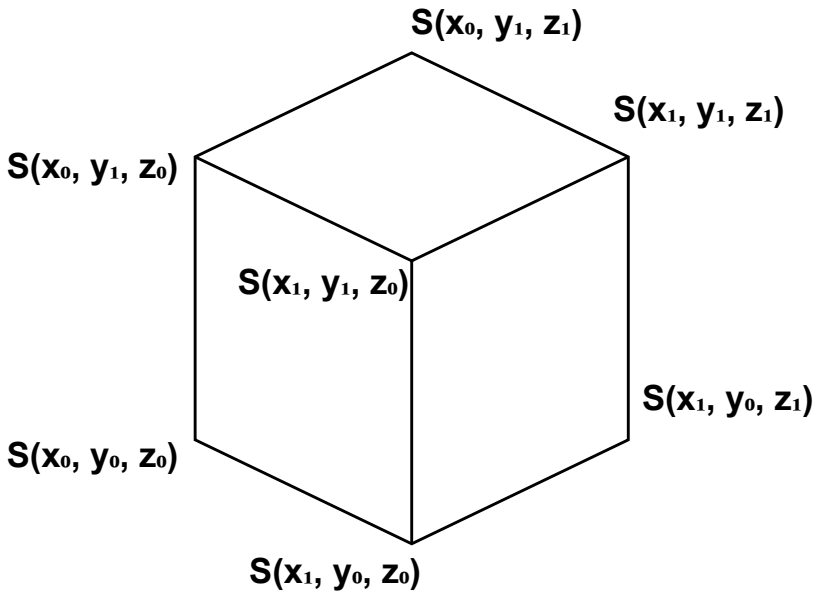
Bourke (1999) kertoo, että vaikka interpolointimenetelmiä on hyvin paljon, graafisissa ohjelmissa käytetään vain verrattain pientä interpolointimenetelmien joukkoa. Tämä johtuu siitä, että graafisissa ohjelmissa joudutaan yleensä interpoloimaan hyvin monta kertaa pienessä ajassa, jolloin laskutehollisesti vaativammat menetelmät eivät ole hyviä vaihtoehtoja. Jotkin yleisesti graafisissa ohjelmissa käytetyt interpolointimenetelmät soveltuvat myös pääosin vain kaksiulotteisiin datajoukkoihin, jonka takia näitä menetelmiä ei voida käyttää tässä tutkielmassa. Han (2013) kertoo, että yleisimmät kolmiulotteisen datan interpolointimenetelmät ovat lähimmän naapurin interpolointi, kolmilineaarinen (eng. trilinear) interpolointi, kolmikuutio (eng. tricubic) interpolointi ja B-mutka interpolointi (eng. B-spline).

Kaufman (2000) kuvailee yksinkertaisimman tavan interpoloida arvoa, olevan lähimmän naapurin interpolointimenetelmä. Lähimmän naapurin interpoloinnissa minkä tahansa $\mathbb{R}^3$ paikan arvo avaruudessa on lähimmän diskreetin pisteen arvo $S(x,y,z)$. Koska tutkielman oletuksena on, että joukon S mittaustulokset ovat mitattu tasaisin välein, tämä muodostaa yhdenmukaisia alueita, joilla vallitsee yksi arvo. Tällöin volumetrista dataa käsitellään käytännössä vokselidatana.

Lähimmän naapurin interpolointia voidaan pitää erityistapauksena luonnollisen naapurin interpolointimenetelmästä. Tässä erityistapauksessa datan mittaustulokset ovat isotrooppisia, kun taas yleisesti ottaen luonnollisen naapurin interpolointi toimii myös hajanaisilla datajoukoilla. Beutel ym. (2011) kertovat kolmiulotteisen datan luonnollisen naapurin interpolointimenetelmän perustuvan siihen, että datan mittaustulosten joukosta S muodostetaan Voronoin diagrammin. Voronoin diagrammi muodostuu alueista nimeltään Voronoin solut, jotka muodostuvat erillisten mittaustulosten $S(x,y,z)$ ympärille. Näiden alueiden rajat muodostuvat sen perusteella, mikä arvo on rajaa lähin. Toisin sanoen, luonnollisen naapurin interpolointi on sama kuin lähimmän interpolointi, mutta se soveltuu myös epäsäännöllisten datojen interpolointiin. Koska oletuksena on, että data on isotrooppista, tutkielmassa voidaan keskittyä vain lähimmän naapurin interpolointiin.

Toinen yleinen interpolointimenetelmä on kolmilineaarinen interpolointi. Csébfalvi (2019) kuvailee menetelmän olevan yleisesti ottaen standardimenetelmä volumetrisen datan visualisointiin. Tämä johtuu siitä, että menetelmä tarjoaa hyvän tasapainon suorituskyvyn ja kuvan laadun välillä. Bourke (1999) kertoo kolmilineaarisen interpoloinnin toimivan niin, että jokainen kolmiulotteisen avaruuden piste on kuution sisällä. Kuvassa 2 esitetty $2 \times 2 \times 2$ hilapisteen rajaaman kuution kulmat ovat 8 lähimmän diskreetin avaruuden pisteet. Tällöin jokaista kuution kulmaa vastaa diskreetin pisteen arvo $S(x,y,z)$. Interpoloitu arvo saadaan tämän kuution avulla, laskemalla kuution muodostavien diskreettien pisteiden arvojen painotettu keskiarvo. Painotus saadaan tässä tapauksessa siitä miten lähellä interpoloitava piste on kyseistä kuution kulmapistettä. Mitä lähempänä, sitä isompi painoarvo kuution kulmaa vastaavalla arvolla $S(x,y,z)$ on. Koska pisteen interpoloitu arvo saatiin pelkästään sen läheisten pisteiden perusteella, kutsutaan kolmilineaarisen interpoloinnin tulosta lokaaliksi arvoksi.

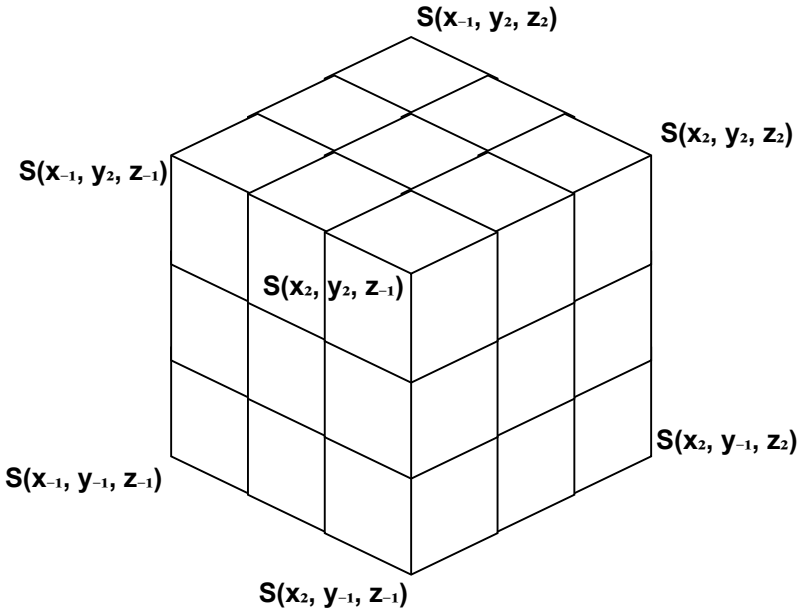
Kolmilineaarista interpolointimenetelmää parempaa kuvan laatua saadaan kolmikuutio inter-



Kuvio 2: Kolmilineaarisen interpoloinnin mittaustulosten muodostama kuutio.

poloinnilla, mutta tämä tulee heikomman suorituskyvyn kustannuksella. Arata (1995) kertoo, että kolmikuutio interpoloinnin edut verrattuna kolmilineaariseen interpolointiin tulevat esille varsinkin silloin, jos interpoloitava data ei ole isotrooppista. Kolmilineaarinen interpolointi voi tuottaa anistrooppisten datojen tapauksissa huomattavia visuaalisia artefakteja. Kolmikuutio interpoloinnissa näiltä artefakteilta suurimmaksi osaksi vältytään. Tämä johtuu siitä, että arvon luomiseen käytetään arvoja suuremmalta lokaalilta alueelta, jolloin lopputulos ottaa paremmin huomioon datan anistrooppisen luonteen. Lekien ja Marsden (2005) kertovat, että kolmikuutio interpoloinnissa interpoloitavaan arvoon $f(x,y,z)$ vaikuttaa 64 lähimmän pisteen arvot. Kuvassa 3 esitetään, kuinka pisteen ympärille muodostetaan kuutio samalla tavalla kuin kolmilineaarisessa interpoloinnissa, mutta kuutio on $4 \times 4 \times 4$ hilapisteen rajoittama. Näin voidaan muodostaa diskreettien pisteiden välisen lineaarisen suoran sijaan korkeamman asteen yhtälön. Tämän yhtälön avulla voidaan arvioida diskreettien pisteiden välillä olevia arvoja.

Arata (1995) selittää, että käytännön toteutuksessa kolmiulotteisen avaruuden arvoja saadaan tehtyä interpoloimalla erikseen jokaisen avaruuden ulottuvuuden suuntaisesti. Ensin interpoloidaan 16 tulosta X-akselin suuntaisesti. Tämän jälkeen näiden 16 tuloksen perusteella interpoloidaan 4 tulosta Y-akselin suuntaisesti. Lopuksi saadaan interpoloinnin arvo



Kuvio 3: Kolmikuutio interpoloinnin mittaustulosten muodostama kuutio.

interpoloimalla aiemmat 4 tulosta Z-akselin suuntaisesti. Arata (1995) tarkentaa kuitenkin, että akselien interpolointijärjestys ei ole oleellinen asia, vaan yksiulotteiset interpoloinnit voidaan teoriassa suorittaa missä järjestyksessä tahansa. Arata (1995) muistuttaa myös, että volumetrisen data-alueen rajojen läheiset tapaukset, joissa johonkin suuntaa pisteitä ei ole tarpeeksi kolmikuutio interpolointia varten, on otettava huomioon jotenkin. Tämä voidaan tehdä esimerkiksi niin, että rajan läheisten pisteiden arvot lasketaan kolmilineaarisella interpoloinnilla.

Lee, Wolberg ja Shin (1997) kertovat B-mutka interpolointimenetelmästä. B-mutka interpolointomenetelmä perustuu yleisesti siihen, että interpoloitavan data-alueen kohdalle muodostetaan kontrollihila (eng. control lattice). Tämän hilan tarkoituksena on muodostaa B-mutka, joka perustuu kontrollihilan alueiden arvoihin. Menetelmässä pyritään yleensä aloittamaan kontrollihilalla, jonka alueet ovat aluksi karkeasti muodostettuja, mutta algoritmin edetessä pyritään tekemään hienojakoisempia kontrollihiloja. Näiden hilojen funktioiden arvot lasketaan yhteen interpoloitavan pisteen (x, y, z) kohdalla, jolloin saadaan pisteen interpoloitu arvo. Lee, Wolberg ja Shin (1997) kertovat B-mutka interpoloinnin olevan erityisen hyvä vaihtoehto interpolointimenetelmäksi silloin, kun volumetrisen datan mittaustulokset ovat hajanaisia tai epäyhtenäisiä. Koska tutkimuksessa oletetaan, että käsiteltävä volumetrinen

data on isotrooppista, B-mutka interpolointi ei ole tässä tapauksessa otollisin menetelmä tutkimuksen käyttötarkoituksiin.

Matechik ja Stytz (1994) kertovat kriging interpolointimenetelmästä. Kriging interpoloinnissa interpoloitava pisteen arvo $f(x,y,z)$ pyritään laskemaan samalla tavalla, kuten kolmilineaarisessa interpoloinnissa. Tämä tarkoittaa sitä, että läheisten diskreetin avaruuden mittaustulosten painotettu keskiarvo määrittää interpoloitavan pisteen arvon. Erona kolmilineaariseen interpolointiin, kriging interpoloinnissa mittaustulosten painoarvo ei tule pelkästään niiden etäisyydestä interpoloitavaan pisteeseen (x,y,z). Matechik ja Stytz (1994) kertovat, että he käyttävät pienimmän neliösumman (eng. least squares) menetelmää painoarvojen laskemiseksi, jotta arviointivirheet pienenisivät. Matechik ja Stytz (1994) mainitsevat, että kriging interpoloinnin vahvuus on se, että se tuottaa hyvin vähän visuaalisia artefakteja luodussa kuvassa. Sen rajoitteena muihin edellä mainittuihin menetelmiin, on kuitenkin se, että sen toteuttaminen on laskennallisesti vaativampaa, jonka takia menetelmän käyttäminen reaaliajassa on hankalaa.

2.3 Volumetrisen datan rakenne

Aiemmassa luvussa käytiin läpi kuinka volumetristä dataa voidaan käsitellä arvoina kolmiulotteisessa hilassa. Luonnollisesti näin löyhälle määritelmälle löytyy myös useita erilaisia tiedostomuotoja, joilla volumetristä dataa voidaan esittää. Koska kolmiulotteisen hilan voi jakaa useaksi kaksiulotteiseksi hilaksi, melkein mitä tahansa kuvien tallentamiseen käytettyä tiedostoformaattia voidaan käyttää volumetrisen datan tallentamiseen, tallentamalla yksittäisiä kaksiulotteisia hiloja erillisiksi kuvatiedostoiksi. Tämä vaatii tosin sen, että käytettävän kuvaformaatin on pystyttävä visualisoimaan kaikkia mahdollisia volumetrisen datan arvoja v. Esimerkiksi Schindelin ym. (2012) kertovat Fiji[1] nimisestä kuvien prosessointiin tarkoitetusta ohjelmasta, jota käytetään yleisesti myös volumetrisen datan käsittelyyn. Schindelin ym. (2012) kertovat Fijin tukevan eri liitännäisten avulla useita eri tiedostomuotoja volumetrisen datan esittämiseen.

Volumetrisen datan tiedostomuotojen väliset oleellisimmat erot ovat eriävät irrallisten tie-

1. https://imagej.net/software/fiji/

dostojen lukumäärät, metatiedot, joiden avulla volumetrisen datan dimensiot voidaan selvittää, tavusyvyys, joka rajoittaa v:n mahdollisia arvoja, sekä mahdollisesti käytettävä pakkausalgoritmi. Eri pakkausalgoritmit yleensä pienentävät tiedostojen tiedostokokoa. Tämä tulee tosin sillä hinnalla, että yksittäisen mittaustuloksen $S(x,y,z)$ hakemisessa saattaa kulua kauemmin. Tämä johtuu siitä, että pakkausalgoritmin purkamiseen kuluu tietokoneelta aikaa, muistia sekä prosessointitehoa. Eri tiedostomuotojen rakenteessa voi olla myös eroja datan osituksesta johtuen. Machiraju ja Yagel (1995) kertovat esimerkiksi, että dataa voidaan osioida tehostamaan tilavuushahmonnuksen toteuttamista datajoukolle.

Goode ym. (2013) kertovat, että yksi olennainen ongelma digitaalisissa poikkileikkauksissa, joilla myös volumetrista dataa esitetään, on se että varsinaista yhtä kaikkien käyttämää tiedostoformaattia ei ole. Tämä johtuu osin siitä, että eri tiedostoformaatit tarjoavat laajalle käyttäjäkunnalle erilaisia ominaisuuksia. Koska tutkielmassa toteutetaan volumetrisen datan vapaavalintaisten poikkileikkausten tarkastelu jo olemassa olevaan ohjelmaan, voidaan kuitenkin keskittyä vain niihin tiedostomuotoihin joita GeoLab tukee. Iivanainen ym. (2021a) kertovat GeoLab-ohjelmiston tukevan kolmea erilaista volumetrisen datan esitysmuotoa, jotka ovat:

- Kuvasekvenssi (eng. image sequence)
- Raakadata
- 3D-TIFF

Kuvasekvenssissä kolmiulotteisen hilan arvot ovat tallennettu erillisiin kuvatiedostoihin kaksiulotteisina hiloina. Koska tässä tutkielmassa keskitytään tutkimaan vain yksikanavaista volumetristä dataa, kuvat ovat mustavalkoisia, eli värikanavia on vain yksi. Yksittäinen kuva edustaa jonkin pääakselin suuntaista volumetrisen datan poikkileikkausta tietystä syvyydestä. Iivanainen ym. (2021a) kertovat, että tuettuja kuvatiedostomuotoja kuvasekvenssille ovat `PNG` ja `TIFF`.

Wiggins ym. (2001) kuvailevat `PNG`-tiedostojen olevan häviämättömästi pakattuja kuvatiedostoja, joita voi lukea niin ylhäältä alas tai oikealta vasemmalle. Wiggins ym. (2001) kertovat `TIFF`-tiedostojen olevan rakenteeltaan monimutkaisempia. `TIFF`-tiedostojen otsikon (eng. header) jälkeen, tiedosto sisältää erillisiä datalohkoja, jotka sisältävät jonkin tunnisteen

(eng. tag). Nämä tunnisteet määrittelevät kuinka kuvatiedostoa pitäisi käsitellä, ja virallisia tunnisteita `TIFF`-tiedostoille on yli 70. Nämä tunnisteet voivat sisältää esimerkiksi tiedon siitä mitä pakkausalgoritmia tiedoston pakkauksessa on käytetty, tai miten kuvan pikselidataa kuuluisi lukea.

Yhteistä molemmille formaateille on se, että ne sisältävät metatiedoissa kuvien bittisyvyyden, eli kuinka monta bittiä vastaa yksittäisen pikselin yhden värikanavan arvoa. Metatiedoista löytyy myös kuvien leveys ja korkeus. Koska yksittäinen kuva on kolmiulotteisen hilan poikkileikkaus yhden pääakselin suuntaisesti, kuvan leveydestä ja korkeudesta saadaan selville kolmiulotteisen hilan kahden pääakselin suuntaiset dimensiot. Kolmiulotteisen hilan kolmannen pääakselin suuntainen dimensio voidaan selvittää laskemalla kuvasekvenssin kuvien määrä, koska yksittäinen kuva on yhden yksikön paksuinen poikkileikkaus kolmiulotteisesta hilasta.

Raakadata esittää joukon S mittaustuloksia v, sisältämällä pelkkiä mittaustulosten arvoja yksi toisensa jälkeen. Mittaustulosten paikka (x, y, z) voidaan määritellä arvojen $v_1, \ldots, v_n$ järjestyksestä, koska dataa käsitellään isotrooppisena. Raakadatassa ei käytetä pakkausalgoritmeja, joten niiden tiedostojen koot ovat yleensä vastaavaa kuvasekvenssiä suurempia. Tällöin myös yksittäisiä mittaustulosten arvoja v on nopeampi lukea, koska tietokoneen ei tarvitse käyttää aikaa pakkausalgoritmin purkamiseen. Suurin ero raakadatan ja kuvasekvenssin välillä on, että raakadata ei sisällä metatietoja. Iivanainen ym. (2021a) selittävät GeoLab-ohjelman ratkaisevan ongelman niin, että käyttäjän on syötettävä raakadatan mittaustulosten bittisyvyys, bittijärjestys sekä volumetrisen datan dimensiot raakadatan lataamisen yhteydessä.

Iivanainen ym. (2021a) kertovat, että 3D-TIFF tarkoittaa yksittäistä `TIFF`-tiedostoa, joka sisältää useita erillisiä poikkileikkauskuvia. Tämä siis on hyvin samankaltainen kuin kuvasekvenssi, mutta usean erillisen kuvatiedoston sijasta kaikki volumetrisen datan poikkileikkaukset löytyvät yhdestä tiedostosta. Koska formaatti ei ole täysin tuettu GeoLabin puolelta, sitä ei ole tarkoitus sisällyttää tutkielmassa tuetuksi volumetrisen datan formaatiksi.

2.4 Volumetrisen datan visualisointi

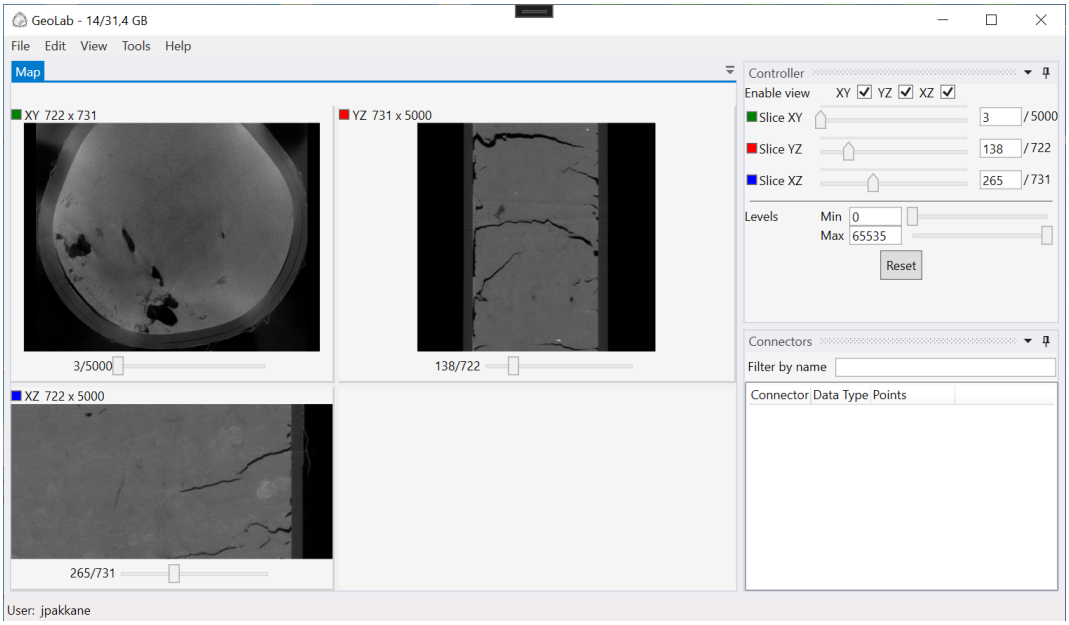
Volumetrisen datan visualisointiin on eri tapoja. Levin ym. (2018) esittävät kolme yleisintä eri tapaa visualisoida volumetristä dataa. Nämä menetelmät ovat tilavuushahmonnus, pintahahmonnus (eng. surface rendering) ja viipalointi. Drebin, Carpenter ja Hanrahan (1988) kuvailevat tilavuushahmonnuksen olevan menetelmä, jolla pystytään visualisoimaan volumetrisen datan kuvaaman kohteen sisällä olevien pisteiden (x,y,z) arvojen v eroja. Levin ym. (2018) kertovat pintahahmonnuksen olevan menetelmä, jolla volumetrisen datan kuvaamasta kohteesta voidaan visualisoida kohteen ulkokuori. Tämä perustuu käyttäjän valitsemaan raja-arvoon, jonka perusteella ulkokuori muodostetaan. Machiraju ja Yagel (1995) kuvailevat viipaloinnin olevan tehokas ja käytännöllinen tapa volumetrisen datan esittämiseen. Viipaloinnissa määritetään kaksiulotteinen taso volumetrisesta datasta käyttäjän haluamasta kohdasta, ja esitetään tason poikkileikkaus kaksiulotteisena kuvana. Tässä tutkielmassa keskitytään toteuttamaan viipalointia volumetrisen datan visualisointimenetelmänä.

Machiraju ja Yagel (1995) kuvailevat viipaloinnin olevan ytimessään prosessi, jolla luodaan uusia arvoja mittaustulosten joukon S arvojen v perusteella (eng. resampling process). Heidän mukaansa viipaloinnissa on etuja tilavuushahmonnukseen verrattuna. Jotta tilavuushahmonnusta voidaan tehdä suurille datajoukoille tehokkaasti, volumetrisen datan sisältävä tiedosto on oltava jaoteltu tai on oltava funktio, joka osaa suoraan muuntaa pisteen (x,y,z) visuaalisesti esitettävään muotoon. Edellä mainittujen tekeminen ei ole kuitenkaan triviaali tehtävä, ja niiden luomiseen menee aikaa. Alaluvussa 2.3 kerrotaan, että yksikään GeoLabin tukema tiedostomuoto ei takaa edellä mainittuja ominaisuuksia. Tämän takia mielivaltainen viipalointi sopii parhaiten ohjelmaan lisättäväksi ominaisuudeksi visualisoida volumetristä dataa.

2.5 GeoLab

Kuvassa 4 esitetty GeoLab on ohjelmisto, johon tutkimuksessa kehitettävä ohjelmistomoduuli integroidaan. Kuvassa 4, GeoLabiin on ladattu GTK:lta saatua dataa. Data on leikattu osa alkuperäisestä datasta, ja sen dimensiot ovat $722 \times 731 \times 5000$. Iivanainen ym. (2021a) kertovat GeoLabin olevan kehitetty Jyväskylän yliopiston sovellusprojektikurssilla GTK:lle

vuonna 2021. Ohjelmiston tarkoitus on auttaa hallinnoimaan multimodaalista tutkimusdataa. Tämä suoritetaan tarjoamalla työkalut kolmiulotteisten harmaasävykuvien, kuten tietokonetomografiakuvien, tarkasteluun sekä antamalla käyttäjille työkalut liittää kolmiulotteisiin kuviin liitedataa. GeoLabissa käyttäjä pystyy myös vaihtamaan näytettävää harmaasävyaluetta, jolloin tietyn arvon ylittävät mittaustulokset näytetään täysin valkoisina, ja toisen arvon alittavat mittaustulokset näytetään täysin mustana. Kaikki mittaustulokset, jotka sijoittuvat näiden kahden arvon välille, skaalataan kattamaan koko näytettävä väriavaruus. Vaikka GeoLab pystyy lukemaan poikkileikkauksia dynaamisesti, sen toiminta on rajoitettu vain pääakselien suuntaisten poikkileikkausten esittämiseen.



Kuvio 4: GeoLab-ohjelmisto.

3 Tutkimusmenetelmä

Tämä luku käy läpi tutkimuksessa sovellettavaa tutkimusmenetelmää sekä sen yhteydessä käytettäviä arviointimenetelmiä. Alaluku 3.1 sisältää tietoa tutkimuksessa käytettävästä tutkimusmenetelmästä sekä miksi tutkimusmenetelmä valittiin käytettäväksi tutkimuksessa. Alaluvuissa 3.2 ja 3.5 keskitytään käsittelemään yleisesti tutkimusmenetelmässä käytettäviä arviointimenetelmien joukkoja, joiden arviointimenetelmiä hyödynnetään tutkimuksessa. Alaluvuissa 3.3, 3.4 ja 3.6 käsitellään tarkemmin kolmea eri arviointimenetelmää, joita käytetään tutkimuksessa. Arviointimenetelmistä käydään läpi miksi niitä käytetään sekä miten niiden käyttöä sovelletaan tutkimuksen yhteydessä.

3.1 Suunnittelutieteellinen tutkimus

Tutkimuksen idea lähti alun perin GeoLab-ohjelman kehityksen yhteydessä. Kehitystyön yhteydessä todettiin, että volumetrisen datan vapaamuotoisempi visualisointi toisi ohjelmalle lisäarvoa. Koska GeoLab on suunniteltu toimivan perustietokoneilla sekä olevan helppokäyttöinen, päädyttiin siihen, että volumetrisen datan viipalointi olisi paras vaihtoehto lähteä lähestymään visualisointia. Koska alan tutkimus on keskittynyt pääosin tilavuushahmonnukseen, viipaloinnista löytyi verrattain vähän tutkimusta, jossa keskityttiin mielivaltaisen viipaloinnin toteuttamiseen todellisessa käyttöympäristössä. Tämän takia tutkimusmenetelmäksi valittiin suunnittelutieteellinen tutkimus, jossa aikaisempia tieteellisissä artikkeleissa määriteltyjä viipalointimenetelmiä toteutetaan todellisessa käyttöympäristössä.

Tutkimuksessa käytetään tutkimusmenetelmänä suunnittelutieteellistä tutkimusta. Hevner ym. (2004) kertovat, että suunnittelutieteellisessä tutkimuksessa pyritään etsimään ratkaisu jollekin ongelmalle, ja tätä ratkaisua kutsutaan tutkimusmenetelmässä nimellä artefakti. Artefaktin luominen ja kehittäminen onkin siis iteratiivinen prosessi, jonka tarkoituksena on lähestyä parasta toistaiseksi löydettyä ratkaisua jokaisella iteraatiolla.

Koska artefakti on oleellinen osa suunnittelutieteellistä tutkimusta, sen määritteleminen on hyvin tärkeää. Tämä määrittely auttaa tutkielmassa todistamaan, että tutkimuksen aihe kontribuoi tutkimusalan tutkimukseen, perustelemaan tutkimuksen toteutuksen perustuvan täs-

mällisiin menetelmiin sekä auttamaan otollisten arviointimenetelmien valitsemisessa. Hevner ym. (2004) jakavat suunnittelutieteelliset artefaktit IT-alalla 4 eri tyyppiin. Konstrukti artefaktit määrittelevät tutkimusaiheen termistöä, merkintätapoja sekä käytettäviä symboleja. Konstruktien tavoite on määrittää ja yhtenäistää sitä, miten tutkittavaa aihetta kuvaillaan ja käsitellään tieteellisessä yhteisössä. Malli artefaktit esittävät tutkittavan ongelman ratkaisuksi abstraktin menetelmän. Tämä menetelmä esitetään aiemmin tutkimusaiheesta tehtyjen konstruktien avulla ja sen pyrkimys on määrittää jokin ratkaisu tutkittavaan ongelmaan. Metodi artefaktit esittävät algoritmin tai toimintatavan jotka ratkaisevat tutkittavan ongelman. Metodit perustuvat aiemmin tutkimusaiheesta tehtyjen tutkimusten malleihin, ja pyrkivät vähentämään näistä malleista abstraktiota ja tekemään mallien ratkaisuista konkreettisempia. Instantiaatio artefakti on ratkaisun toteutus tai demonstraatio, ongelman oikeassa käyttö- tai esiintymisympäristössä. Instantiaatiot perustuvat aiempien tutkimusten metodeihin ja pyrkivät todistamaan, että metodit toimivat käytännössä.

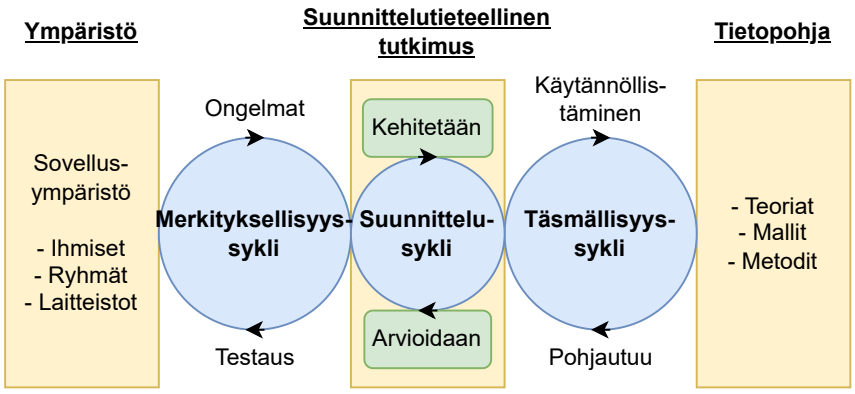
Iivari (2007) ehdottaa vaihtoehtoista jakotapaa artefaktityypeille, joka perustuu artefaktin tarjoamaan toiminnallisuuteen. Tässä jakotavassa artefakti voi kuulua myös useaan eri arkkityyppiin samanaikaisesti. Iivari (2007) väittää, että artefaktien yhteydessä ihmisiä yleensä kiinnostaa eniten se, mitä artefakti tekee, jolloin myös artefaktien jako pitäisi tehdä tästä näkökulmasta. Iivari (2007) listaa arkkityyppien jaottuvan automatisointiin, arvon lisäämiseen (eng. to augment), asian tai tiedon välittämiseen, tiedottamiseen, viihdyttämiseen, taiteellistamiseen ja kanssa olemiseen.

Mielestäni niin Hevner ym. (2004) kuin Iivari (2007) antavat hyvät perustelut artefaktityyppien jaotteluun. Olen kumminkin sitä mieltä, että Hevner ym. (2004) luoma artefaktityyppien jako on selkeämpi käyttää tämän tutkielman yhteydessä. Tämä perustuu siihen, että tässä tutkielmassa luodaan heidän jaon mukaan selkeästi instantiaatiota. Iivari (2007) tekemän jaottelun mukaan tutkimuksessa luotu ohjelmistomoduuli voitaisiin katsoa tarjoavan käyttäjälleen arvon lisäämistä, tarjoamalla helppokäyttöisen käyttöliittymän, jonka avulla volumetrisen datan tarkastelu on helpompaa. Mielestäni tämä määritelmä yksistään ei kuitenkaan tuo esille ohjelmistomoduulin interpolointimenetelmän osuutta tutkimuksessa, joka on olennainen osa ohjelmistomoduulin toimintaa. Jaottelusta puuttuu mielestäni toiminto, joka kuvaisi volumetrisen datan interpoloinnin kykyä jalostaa dataa ihmisten ymmärrettäväksi tiedoksi,

joka on yksi tämän tutkimuksen päätavoitteista.

Niin Hevner ym. (2004) kuin Iivari (2007) ovat kuitenkin yhtä mieltä siitä, että suunnittelutieteellinen tutkimus koostuu kolmesta syklistä. Näitä kolmea sykliä havainnollistetaan kuviossa 5. Kuviossa 5 esitetyistä sykleistä keskimmäinen on toteutus-arviointi-iteraatiot, jotka muodostavat suunnittelutieteellisen tutkimuksen ytimen. Näiden lisäksi on myös merkityksellisyyssykli (eng. relevance cycle) ja täsmällisyyssykli (eng. rigor cycle). Näiden syklien ideana on luoda puitteet suunnittelutieteellisen tutkimuksen tekemiselle.

Merkityksellisyyssyklissä ongelman esiintymisympäristössä työskentelevät henkilöt ja organisaatiot määrittelevät ongelmia ja mahdollisuuksia, joista syntyvät vaatimukset, jotka muodostavat suunnittelutieteellisen tutkimuksen lähtökohdan. Ilman näitä vaatimuksia, suunnittelutieteellinen tutkimus ei ole merkityksellistä aihepiirin suhteen, jolloin tutkimukselta häviää pohja. Valmistuttuaan suunnittelutieteellisessä tutkimuksessa valmistunutta artefaktia päästään myös kenttä testaamaan ongelman käyttöympäristössä. Täsmällisyyssykli taas muodostaa rajapinnan, jonka avulla suunnittelutieteellinen tutkimus käyttää ja muokkaa tutkimusalalla olevaa tietopohjaa. Suunnittelutieteellisessä tutkimuksessa artefakti pyritään luomaan nojautuen olemassa olevaan tietopohjaan. Tämä taas auttaa havainnollistamaan, että tietopohjan muodostavat teoriat, metodit, mallit ja muut puitteet toimivat käytännössä. Tämän lisäksi suunnittelutieteellinen tutkimus saattaa myös lisätä uutta tietoa tietopohjaan tutkimuksen myötä.



Kuvio 5: Kaavio havainnollistamaan suunnittelutieteellisen tutkimuksen kolmea sykliä (mukaillen Hevner 2007).

Hevner ym. (2004) kertovat, että hyvän suunnittelutieteellisen tutkimuksen pitäisi täyttää

seuraavat listatut seitsemän kriteeriä, ja tutkielmassa tulisi myös avoimesti tuoda esille miten nämä kriteerit täyttyvät.

1. Tutkimuksessa artefaktin on oltava selkeästi ja hyvin määritelty.
2. Ratkaistavan ongelman tutkiminen on olennaista tutkittavan aihepiirin osalta.
3. Artefaktin toiminta on osoitettava täsmällisesti arviointimenetelmien avulla.
4. Tutkimus kontribuoi jotakin uutta tieteen alalle.
5. Tutkimuksen toteutus- ja arviointivaiheet perustuvat täsmällisiin menetelmiin.
6. Suunnittelutiede on etsintäprosessi, eli ongelman parasta ratkaisua lähestytään iteratiivisesti.
7. Tutkimuksen tulokset tuodaan selkeästi sekä yksityiskohtaisesti esille.

Iivari (2007) tarjoaa myös oman näkemyksensä suunnittelutieteistä, joka tarkentaa ja antaa vaihtoehtoisia näkemyksiä osalle listan kriteereistä. Hän myös kiteyttää ideansa 12 teesiin. Hevner (2007) kertoo, että hän käytännössä on hyvin pitkälti samaa mieltä Iivarin idealistauksen kanssa. Hevner ym. (2004) luoman jaottelu tarjoaa kuitenkin konkreettisemmat puitteet tarkastella tutkimusta, joten seuraavissa kappaleissa käsitellään, kuinka tämä tutkimus täyttää Hevnerin listaamat kriteerit. Iivari (2007) tuo kuitenkin esille useita mielestäni tärkeitä yksityiskohtia suunnittelutieteellisen tutkimuksen toteuttamiseen, joten hänen näkemyksiään tuodaan myös esille niissä kohdissa, joissa se on mielestäni oleellista.

Hevner ym. (2004) kuvailevat, että suunnittelutieteellisessä tutkimuksessa artefakti on olennainen osa tutkimusmenetelmää. Tämän takia onkin tärkeää määritellä artefakti tarkasti, jotta tutkimuksen muut olennaiset osat voidaan määritellä. Tutkimuksessa suunnitellaan ja kehitetään ohjelmistomoduuli joka perustuu aiempiin volumetrisen datan visualisoinnin tutkimusten määrittämiin metodeihin. Tämä moduuli myös liitetään olemassa olevaan volumetristä dataa käsittelevään ohjelmistoon, eli ongelman ratkaisun oikeaan käyttöympäristöön. Kehitettävän ohjelmistomoduulin artefaktin tyyppi on siis instantiaatio.

Hevner ym. (2004) kertovat, että tutkimuksessa ratkaistava ongelma ei voi olla merkityksetön, jotta sen ratkaiseminen tuottaisi yleisesti hyötyä tutkittavalla tieteenalalla. Luvussa 2 käydään tarkemmin esimerkkien kanssa läpi, mitä hyötyjä suurikokoisen volumetrisen datan visualisoinnista on. Yleisesti ottaen volumetrisen datan visualisointi auttaa antamaan ihmi-

sille kuvaa volumetrisen datan kokonaisuudesta, ja volumetristä dataa saadaan yleensä eri tieteenaloilla erilaisten mittauslaitteiden mittaustuloksista.

Hevner ym. (2004) kertovat, että toteutettavan artefakti toiminnan varmistaminen täsmällisillä arviointimenetelmillä, on olennainen osa suunnittelutieteellistä tutkimusta. Alaluvuissa 3.3, 3.4 ja 3.6 käydään läpi arviointimenetelmiä, joita tutkimuksessa käytetään artefaktin arviointiin. Luvussa 5 käydään läpi eri iteraatioiden erillisiä arviointeja. Tutkimuksessa käytetään niin analyyttisiä kuin kuvailevia arviointimenetelmiä artefaktin eri iteraatioiden arviointiin. Näiden lisäksi, koska tutkimus suoritetaan yhteistyössä GTK:n kanssa, heiltä saadaan eri iteraatioista palautetta. Tämä palaute otetaan arvioinneissa huomioon, sillä se on oleellinen osa kuviossa 5 kuvailtua merkityksellisyyssykliä. Tällä tavoin tutkimuksessa päästään toteuttamaan kommunikointia sovellusympäristön kanssa tutkimuksen edetessä.

Hevner ym. (2004) kuvailevat tärkeäksi sen, että tietotekniikan suunnittelutieteellinen tutkimus kontribuoi tutkimusaiheeseen ja se myös osoitetaan tutkimuksessa. Tämä tarkoittaa, että tutkimuksen aihe käsittelee joko uutta ongelmaa, tai jo ratkaistua ongelmaa käsitellään uuden artefaktin näkökulmasta. Vaikka suurikokoisen volumetrisen datan visualisointia käsittelevää tutkimusta on jo olemassa, tutkimus keskittyy suuriltaosin tilavuushahmonnukseen. Vaikka myös tutkimuksessa käytettävää mielivaltaista poikkileikkaamista käsittelevää tutkimusta on jo olemassa, tutkimuksissa ei ratkaista ongelmaa sen oikeassa esiintymisympäristössä.

Hevner ym. (2004) kertovat, että niin suunnittelutieteellisessä kuin käyttäytymistieteellisessä tutkimuksessa on tärkeää, että tutkimusta tehdään täsmällisesti. Tämä tarkoittaa että tutkimuksessa tehtyjä ratkaisuja voidaan perustella pohjautuen tutkittavan aihealueen valmiin tietopohjan avulla. Iivari (2007) kertoo olevansa eri mieltä Hevnerin ym. kanssa siitä mikä määrittelee sen, että tutkimus on toteutettu täsmällisesti. Iivari (2007) määritteleekin täsmällisesti toteutetun tutkimuksen olevan tutkimusta, jossa tuodaan tutkimuksen vaiheet ja lähtökohdat hyvin selkeästi esille. Iivari (2007) painottaa myös, että täsmällinen lähestymistapa erottaa suunnittelutieteellisen tutkimuksen pelkkien IT-artefaktien luomisesta. Artefaktin kehittämistä käsitellään tarkemmin luvussa 4. Kehitysprosessissa kehitysratkaisut pohjautuvat aiempien ohjelmien yleisiin ja hyväksi todettuihin ratkaisuihin sekä aiempien tutkimusten tutkimustuloksiin. Luvussa 2 tuodaan myös esille jo valmiiksi aihepiiriä käsitteleviä ohjelmia sekä mitä tutkimusaiheesta tiedetään jo ennestään.

Hevner ym. (2004) täsmentävät suunnittelutieteellisen tutkimuksen olevan pohjimmiltaan etsintäprosessi, jossa johonkin ongelmaan pyritään etsimään paras tähän mennessä löydetty ratkaisu. Tämä tarkoittaa että tutkimuksessa pyritään olemassa olevan tietopohjan avulla tekemään hyvä ratkaisu, jota arvioidaan. Tämän jälkeen arvioinnin tuloksiin pohjautuen ratkaisua pyritään parantelemaan, ja useiden iteraatioiden jälkeen pyritään pääsemään ja osoittamaan arviointimenetelmillä, että tutkimuksen artefakti on jollain tavalla paras löydetty ratkaisu tutkimuksen käsittelemään ongelmaan. Tutkielman luvuissa 4 ja 5 käydään läpi tutkimuksessa tehtyjä iteraatioita. Tutkielman aikana toteutettiin 3 kehitysiteraatiota, joita jokaista vastaa yksi arviointi-iteraatio.

Hevner ym. (2004) painottavat tärkeäksi sen, että tutkielmassa tuodaan selkeästi esille tutkimuksen tulokset. Tämä tarkoittaa sitä, että tutkimuksessa tuodaan yleisellä tasolla ratkaisu ongelmiin esille, niin että myös henkilö joka ei ole aihepiirin asiantuntija, pystyy ymmärtämään mitä tutkimuksessa tehtiin ja mihin ratkaisuun päädyttiin. Samaan aikaan tutkimuksessa on tuotava ratkaisun yksityiskohdat esille niin tarkasti, että aihepiirin asiantuntija pystyy tarvittaessa toistamaan tai soveltamaan ratkaisua tutkielmaan pohjautuen. Tutkielmassa toteutettu lähdekoodi on julkisesti saatavilla, ja luvussa 7 käydään lyhyesti läpi tutkimuksen tulokset.

3.2 Kuvailevat arviointimenetelmät

Hevner ym. (2004) määrittelevät yhdeksi tietotekniikassa käytettäväksi arviointimenetelmien tyypiksi kuvailevat arviointimenetelmät. Hevner ym. (2004) jaottelevat kuvaileviksi arviointimenetelmiksi tietoon perustuvan argumentin muodostamisen ja skenaarion luomisen. Tietoon perustuvan argumentin muodostamisessa käytetään aiemmista tutkimuksista syntynyttä tietopohjaa hyväksi, jotta voidaan tehdä argumentti, joka puoltaa kehitettävän artefaktin toimintaa. Skenaarion luomisessa on tarkoitus arvioida artefaktin toimintaa hahmotelmalla sen toimintaa jossakin tilanteessa. Esimerkkinä skenaarion luomisesta ovat käyttötapauksien luominen.

Tässä tutkielmassa näkymää ja sen ohjausta arvioidaan tietoon perustuvan argumentin muodostamisella. Tällöin arvioinnissa pyritään perustelemaan näkymän ja sen ohjauksen toimin-

toja aiempien tutkimusten tuloksien avulla. Tämä voidaan toteuttaa joko niin, että aiempi tutkimus osoittaa, että kehitysratkaisu on hyvä tai huono, tai voidaan käyttää aiemmassa tutkimuksessa otolliseksi osoittautunutta arviointimenetelmää, jonka avulla voidaan osoittaa onko jokin toteutusratkaisu toimiva vai ei. Jeffries ym. (1991) tuovat esille neljä metodia käytettävyyden arviointiin. Nämä metodit ovat ohjelmiston toiminnan ohjeistuksien läpikäynti, kognitiivinen läpikäynti, käytettävyystestaus ja heuristinen arviointi.

Jeffries ym. (1991) kertovat, että ohjelmiston toiminnan ohjeistuksia seuraamalla pyritään vertaamaan ohjelman toimintaa tutkimusalalla tehtyihin ohjeistuksiin siitä miten ohjelmiston tulisi toimia. Jos vertauksessa tulee ilmi, että ohjelmiston toiminta rikkoo ohjelmistokehityksen ohjeistuksia, on kyse yleensä ongelmasta. Löydettyjen ongelmien, tai niiden puutteiden avulla voidaan luoda argumentti, joka joko puoltaa tai vastustaa kehitetyn ohjelmiston toimivuutta.

Jeffries ym. (1991) kuvailevat, että kognitiivisessa läpikäynnissä ideana on käydä läpi ohjelman ydintoiminto ottaen huomioon keskiverto käyttäjällä olevat tiedot ohjelman toiminnasta. Tämän jälkeen kontekstissa olevan tiedon avulla pyritään toteuttamaan jokin ohjelman ydintoiminto ja samalla kirjata ylös käyttäjän toiminnot yksityiskohtaisesti. Tällä pyritään tuomaan esille mahdollisia ongelmia, joita voisi tulla vastaan keskiverto käyttäjällä. Jos ongelmia ei ilmene, se puoltaa sitä, että ohjelmisto toimii hyvin.

Jeffries ym. (1991) kuvailevat käytettävyystestauksen olevan arviointimenetelmä, jossa ohjelmisto annetaan koehenkilöille ja heille annetaan jokin tehtävä joka heidän pitää suorittaa ohjelmiston avulla. Koetehtävät suunnitellaan yleensä niin, että niiden suorittaminen onnistuu ohjelmiston perustoimintojen avulla. Tällöin, jos koehenkilöt kohtaavat joitakin ongelmia tehtävän suorittamisessa, on todennäköistä, että kyseessä on jokin ongelma ohjelmiston käytettävyydessä. Löydetyt ongelmat tai niiden puutteet joko puoltavat tai vastustavat argumenttia ohjelmiston toimivuudesta.

Jeffries ym. (1991) kertovat, että heuristisessa arvioinnissa alan asiantuntijat pyrkivät löytämään käytettävyyden ongelmia. Tämän apuna käytetään yleensä jotakin heuristiikkalistaa, joka toimii ohjenuorana, mistä eri näkökulmista ohjelman toimintaa tulisi tarkastella. Myös tässä arviointimenetelmässä, ongelmien tai niiden puutteiden avulla voidaan luoda argument-

ti, joka joko puoltaa, tai vastustaa kehitetyn ohjelmiston toimivuutta.

3.3 Heuristinen arviointi

Jokaisessa aiemmin mainitussa arviointimenetelmässä on niin hyvät kuin huonot puolensa. Ohjelmiston toiminnan ohjeistuksien seuraaminen perustuu vahvasti siihen, että löydetään sopivat ohjeistukset. Smith ja Mosier (1986) tarjoavat esimerkiksi yleisellä tasolla hyvin yksityiskohtaisen ohjeistuksen siitä miten käyttöliittymän tulisi toimia. Tämä ohjeistus sisältää esimerkiksi 199 ohjetta siitä miten ohjelmiston pitäisi toimia käyttäjän syöttäessä siihen dataa. Ottaen huomioon miten yksinkertaisen kokonaisuuden näkymä ja sen ohjaus muodostavat, onkin vaikea löytää ohjeistusta, jota voitaisiin käyttää kokonaisuuden arviointiin tehokkaasti. Tämä johtuu siitä, että ohjeistukset pyrkivät kattamaan monimuotoisia ja valmiita ohjelmia, eikä pelkästään yhteen toimintoon keskittyneitä ohjelmistomoduuleja.

Kognitiivisen läpikäynnin käyttämisessä piilee sama ongelma kuin ohjelman ohjeistuksien seuraamisessa. Arvioitava ohjelmistomoduuli ei muodosta tarpeeksi isoa kokonaisuutta, jolloin myös kognitiivinen läpikäynti jäisi hyvin pienenmuotoiseksi. Tällöin myös siitä saatavat tulokset jäisivät vähäisiksi.

Käytettävyystestaus taas vaatisi ainakin muutaman henkilön, joilla testaus suoritetaan, jotta sillä saaduilla tuloksilla olisi merkittävää arvoa. Nielsen ja Landauer (1993) ja Virzi (1990, 1992) kertovat käytettävyystestauksen vaativan noin 4-5 koehenkilöä, jotta olemassa olevista käytettävyyden ongelmista havaittaisiin noin 85%. Myöhemmissä tutkimuksissa on myös havaittu, että viidellä koehenkilöllä toteutettu käytettävyystestaus voi jäädä hyvinkin kauas aiempien tutkimusten kuvaillusta käytettävyysongelmien löytämisen kattavuudesta. Spool ja Schroeder (2001) havaitsivat, että viisi henkilöä löysivät, hänen tutkimuksessaan, keskimäärin n. 35% käytettävyyden ongelmista. Käytettävyystestauksen suorittaminen vaatii siis mittavia resursseja jotta se voitaisiin toteuttaa. Täten käytettävyystestausta ei ole myöskään varteenotettava vaihtoehto käytettävyyden arviointiin pro gradu -tutkielmaksi mitoitetun tutkielman tapauksessa.

Myös heuristisessa arvioinnissa on omat ongelmansa. Yleensä heuristinen arviointi vaatii useita arvioijia, jotta tulokset ovat luotettavia. Nielsen (1992) toteaakin, että hänen kokemuk-

sien mukaan yksittäinen arvioija löytää noin 35% käytettävyyden ongelmista. Tämän takia olisi tärkeää, että käytettävyyden heuristinen arviointi suoritettaisiin n. 3-5 henkilöllä. Nielsen (1992) mainitsee tosin, että nämä luvut perustuvat hänen kuuteen viimeisimpään projektiin, ja ovatkin yleisellä tasolla vain suuntaa-antavia. Nielsen (1992) kertoo, että tarkemman arvion saamiseksi optimaalisesta arvioijien määrästä projektissa, vaatisi useiden asioiden kuten yksittäisen virheen hinnan, yksittäisen arvioijan hinnan ja arvioijien löytämien virheiden prosentuaalisen määrän arvioinnin. Koska tämä on vain yksi arviointi-iteraatio pro gradu -tutkielmaksi mitoitetussa tutkimuksessa, on hankalaa lähteä arvioimaan edellä mainittuja asioita. Myös useiden arvioijien hankkiminen tässä kontekstissa on rajallisten resurssien takia hankalaa, joten tutkimuksessa joudutaan tyytymään siihen, että heuristista arviointia suoritetaan vain yhden henkilön toimesta.

Heuristisen arvioinnin käyttämistä edesauttaa se, että näkymä ja ohjaus muodostavat suhteellisen pienen kokonaisuuden. Koska tarkasteltava kokonaisuus on pieni, sen arviointi on helpompaa, vaikka arviointia suorittaa vähemmän kokenut henkilö. Tämän takia näkymän ja sen ohjauksen arviointi toteutetaan tarkastelemalla toteutetun ohjelmistomoduulin toimintaa heuristiikkalistojen avulla.

Heuristisessa arvioinnissa käytetään usein apuna listaa, joka auttaa arvioijaa kiinnittämään huomiota oleellisiin asioihin. Heuristiikkalistauksia on useita. Nielsen (1994) kuvailema heuristiikkalistaus on yksi käytetyimmistä heuristiikkalistauksista. Nielsen (1994) kertoo, että hänen muodostama lista koostuu kymmenestä eri periaatteesta, joiden näkökulmasta ohjelman toimintaa tulisi arvioida. Nämä näkökulmat ovat:

1. Ohjelma näyttää koko ajan käyttäjälle mikä sen tila on.
2. Ohjelma käyttää oikean maailman termejä.
3. Käyttämisen hallinta ja vapaus.
4. Ohjelma käyttää termejä ja toimii yhtenäisesti.
5. Virhetilanteiden estot.
6. Tunnistaminen muistamisen sijaan.
7. Joustavuus ja tehokäyttö.
8. Estetiikka ja minimalismi.
9. Virhetilanteiden tunnistaminen, diagnosoiminen ja niistä palautuminen.

10. Avun ja dokumentoinnin tarjoaminen.

Riippumatta siitä mitä menetelmää käytetään ongelmien etsimiseen ohjelmasta, löydetyt ongelmat on yleensä tapana luokitella ongelman vakavuuden mukaan. Luokittelu tehdään siksi, että yleensä kehityksellä on rajalliset resurssit. Tällöin kaikkia löydettyjä ongelmia ei välttämättä pystytä korjaamaan. Tällöin onkin tärkeää priorisoida oleellisimmat ongelmat, jotta kehitykseen varatut resurssit voidaan käyttää tehokkaasti. Ongelmien luokittelumenetelmiä on useita eri tarkoituksiin. Nielsen (1992) käyttää tutkimuksessaan kaksitasoista ongelmien luokittelua, joissa ongelmat ovat jaettu vähäisiin ja vakaviin ongelmiin. Tämä asteikko soveltuu hyvin aikaisessa vaiheessa olevan ohjelmistomoduulin arviointiin, jossa hienojakoisempi jaottelu olisi hankalaa.

Heuristisen arvioinnin puutteita paikataan osin myös sillä, että GTK:lta saadaan ohjelman toiminnasta palautetta. Tällöin kokonaisuudessa heuristisen arvioinnin ja palautteen pitäisi löytää kohtuullinen määrä käytettävyyden ongelmista. Tällöin jahka mahdollisesti löydetyt käytettävyyden ongelmat on korjattu, on mahdollista antaa kohtuullisen hyvän argumentin siitä, että näkymä ja sen ohjaus toimivat hyvin ja toteutuksessa voidaan edetä volumetrisen datan interpolointiin.

3.4 Kognitiivinen läpikäynti

Toteutetun ohjelmistomoduulin integroinnin arvioinnissa käytetään myös kuvailevaa arviointimenetelmää. Tässä kohtaa valmista ohjelmistomoduulia on arvioitu kahdesti omana kokonaisuutenaan. Kolmannen arviointikerran tarkoituksena on arvioida miten helposti käyttäjä pääsee käyttämään toteutettua moduulia GeoLab-ohjelmiston kautta. Kyseessä on siis käytettävyyteen liittyvä arviointi. Tällöin kuvailevat arviointimenetelmät tarjoavat hyviä menetelmiä testata moduulin käytettävyyttä osana GeoLab-ohjelmistoa. Aiemmin läpikäydyistä arviointimenetelmistä käytettävyystestaus ja kognitiivinen läpikäynti tarjoavat hyviä menetelmiä arvioida moduulin integroinnin käytettävyyttä käyttäjän näkökulmasta. Koska käytettävyystestaus vaatii huomattavan määrän resursseja, tässä tutkimuksessa valittiin käytettäväksi arviointimenetelmäksi kognitiivinen läpikäynti.

Mahatody, Sagar ja Kolski (2010) kertovat, että kognitiivisen läpikäynnin toteuttamiseen

on vuosien varrella tehty useita eri metodeja. He listaavat tutkimuksessaan 11 eri metodia. Tässä tutkielmassa käytetään kognitiivisen läpikäynnin kolmatta esitettyä versiota. Vaikka kolmannesta versiosta on tehty useita variaatioita, jotka pyrkivät korjaamaan sen puutteita, jotkin variaatiot, kuten aktiviteetti läpikäynti (eng. activity walkthrough), vaativat lisää aikaa metodin toteuttamiseen. Toiset menetelmät, kuten suoraviivainen kognitiivinen läpikäynti pyrkivät vähentämään kolmannen version työmäärää, mutta keskittyvät myös korjaamaan testaavan ryhmän sisäistä toimintaa. Koska tämä testaus suoritetaan vain yhdellä henkilöllä, variaation parannukset eivät pääse hyvin esille tässä tapauksessa.

Wharton (1994) kertoo, että kognitiivisen läpikäynnin kolmannessa versiossa on määriteltävä selkeä ohjelman tila, johon toimintojen kautta pyritään pääsemään. Tämän lisäksi jokaisen ohjelman tavoitetta kohti vievän toiminnon yhteydessä testaajan on kysyttävä seuraavat neljä kysymystä:

1. Toimiiko käyttäjä oikein saadakseen tavoitteena olevan toiminnon suoritettua?
2. Pystyykö käyttäjä löytämään seuraavaa toimintoa?
3. Pystyykö käyttäjä erottamaan seuraavan toiminnon muista toiminnoista?
4. Saako käyttäjä vastetta jos hän valitsee oikean toiminnon?

Alaluvussa 5.3 käydään tarkemmin läpi, miten kognitiivinen läpikäynti on suoritettu, ja mitkä sen tulokset ovat. Alaluvussa käydään läpi mitä kognitiivisessa läpikäynnissä pyritään saamaan ohjelmalla tehtyä sekä mitkä toiminnot käyttäjän tarvitsee käydä läpi. Tämän lisäksi alaluvussa vastataan yllä esitettyihin neljään kysymykseen jokaisen suoritetun toiminnon yhteydessä sekä lopuksi käydään läpi, mitä mahdollisia käytettävyysongelmia löydettiin.

Kuten kahdessa aiemmassa arvioinnissa, saatuja tuloksia yhdistetään GTK:lta saadun palautteen kanssa. Yhdistelemällä heiltä saatua palautetta ja kognitiivista läpikäyntiä, pyritään saamaan hyvä arvio siitä, että toteutettu ohjelmistomoduuli ja sen toiminto integroidussa ohjelmistoympäristössä on käytettävyydeltään hyvä.

3.5 Analyyttiset arviointimenetelmät

Hevner ym. (2004) määrittelevät analyyttisten arviointimenetelmien muodostavan tietotekniikan tutkimusalalla yhden arviointimenetelmien tyypin. Hevner ym. (2004) luettelevat analyyttisiksi arviointimenetelmiksi staattisen analyysin, arkkitehtuurin analyysin, optimoinnin ja dynaamisen analyysin. Staattisessa analyysissä pyritään tarkastelemaan artefaktista sen muuttumattomia ominaisuuksia, esimerkiksi toteutetun ohjelman lähdekoodin monimutkaisuutta. Arkkitehtuurin analyysissä pyritään tarkastelemaan, miten artefakti toteuttaa ohjelmistoarkkitehtuureiden määrittelemiä ohjelmistorakenteita. Optimoinnissa pyritään joko osoittamaan, että jokin artefaktin ominaisuus on optimaalinen tai artefaktin toiminnalle pyritään etsimään optimaaliset rajat. Dynaamisessa analyysissä arvioidaan artefaktin dynaamisia, eli olosuhteista riippuvia ominaisuuksia, kuten toteutetun ohjelman jonkin ominaisuuden suorituskykyä tietyissä olosuhteissa.

Kaikki edellisessä kappaleessa mainitut menetelmät tarjoavat työkalut interpolointimenetelmien suorituskyvyn arviointiin. Näistä menetelmistä vain dynaaminen analyysi tarjoaa myös mahdollisuuden arvioida interpolointimenetelmien lopputuloksen, eli poikkileikkauskuvan laatua visuaalisesti, joka on olennainen osa interpolointimenetelmien toimintaa. Ball (1999) kertoo, että koska ohjelmaa konkreettisesti suoritetaan, dynaaminen analyysi tarjoaa hyvin tarkat mittarit mitattaville ominaisuudelle. Tällöin saadut tulokset ovat mitattuja, eivätkä arvioituja. Dynaaminen analyysi tarjoaa myös hyvän ympäristön arvioida laajoja algoritmeja, koska sen toteuttaminen vaati vain ohjelman suorittamista, eikä algoritmeja tarvitse käydä käsin läpi.

Dynaamisessa analyysissä on tosin myös heikkoutensa. Ball (1999) kuvailee dynaamisen analyysin heikkouden olevan se, että se perustuu johonkin tiettyyn käytettävään syötteeseen. Tällöin dynaaminen analyysi ei yleensä tarjoa niin laajaa ohjelmiston läpikäynnin kattavuutta kuin muut menetelmät, joilla kaikki ohjelman suoritettavat vaihtoehdot voidaan käydä läpi. Ball (1999) kuvaileekin, että yleensä staattista ja dynaamista analyysiä käytetään rinnakkain, koska ne paikkaavat toistensa puutteita tehokkaasti.

Vaikka dynaaminen analyysi tarjoaa hyvät työkalut tutkielmassa käytettyjen interpolointimenetelmien suorituskyvyn ja kuvanlaadun mittaamiseen, se ei yksinään riitä. Vokolos ja

Weyuker (1998) kertovatkin että yksi tapa saada suorituskykytestauksen testituloksista konkreettisia tuloksia on eri testitulosten vertailu suorituskykyvertailussa (eng. benchmarking). Ainoa vaihtoehto minkä kesken vertailua voidaan tehdä, on ohjelmistomoduulin eri interpolointimenetelmien toteutukset. Tämä johtuu siitä, että ei ole tiedossa aiempaa toteutusta, jossa poikkileikkauksia luetaan dynaamisesti massamuistista samalla tavalla kuin tutkielmassa toteutetussa ratkaisussa.

3.6 Dynaaminen analyysi

Eri interpolointimenetelmien suorituskyvystä ja tuotetusta kuvanlaadusta voidaan tehdä hyviä arvauksia perustuen aiempien tutkimusten tuloksiin sekä luvussa 4.3 esitettyihin interpolointimenetelmien toteutuksissa käytettyihin matemaattisiin esityksiin. Esimerkiksi Han (2013) kertoo, että kuvien interpoloinnissa lähimmän naapurin interpoloinnin olevan noin 20% nopeampi kuin kolmilineaarinen interpolointi ja n. 60% nopeampi kuin kolmikuutio interpolointi. Toisaalta samassa tutkielmassa kerrotaan myös, että interpolointimenetelmien tuottama visuaalinen kuvanlaatu oli käänteinen menetelmien suorituskyvyn kanssa. Myös Fadnavis (2014) kertovat, että heidän tutkimuksessa kuvien visuaalisessa laadussa nähdään samanlainen trendi, jossa hitaammat menetelmät tuottavat parempaa kuvanlaatua.

Tässä arviointi-iteraatiossa ei olla kiinnostuneita tarkoista suorituskyvyn tuloksista. Tämän arvioinnin tarkoitus on varmistaa, että kaikki kolme toteutettua interpolointimenetelmää tarjoavat käyttäjälleen kohtuullisia kompromisseja kuvanlaadun ja suorituskyvyn välillä myös tämän ohjelmistomoduulin ympäristössä, jossa dataa luetaan dynaamisesti massamuistista. Jos jokin interpolointimenetelmistä ei tarjoa muihin menetelmiin verrattavissa olevaa kompromissia suorituskyvyn ja kuvanlaadun välillä, tämä menetelmä poistetaan ohjelmistomoduulin käytettävistä vaihtoehdoista. Tämä arviointi auttaa myös antamaan suuntaa-antavia suorituskyvyn arvoja GeoLabin ohjeisiin. Tällöin käyttäjän on helpompi arvioida, mikä interpolointimenetelmä sopii hänen tarkoituksiinsa parhaiten.

Koska arviointi-iteraation kiinnostuksen kohteena on suuntaa-antavasta arvio interpolointimenetelmien välisestä suorituskyvystä, koodin täyden kattavuuden arviointi ei ole tarpeellista, joka vähentää tarvetta tehdä staattista analyysiä täydentämään dynaamista analyysiä.

Dynaamisen analyysin avulla, jopa pienillä resursseilla voidaan saada jonkinlainen kuva interpolointimenetelmien toiminnasta ohjelmistomoduulin ympäristössä. Muut menetelmät vaatisivat enemmän resursseja, jolloin niitä ei voida käyttää pro gradu -tutkielman yksittäisessä arviointi-iteraatiossa. Myös suorituskyvyltään heikompien interpolointimenetelmien mahdollisesti paremman laadun tuottamat kuvat pääsevät dynaamisessa arviointimenetelmässä arviointikriteeriksi, joka puoltaa vahvasti dynaamisen analyysin käyttöä arviointiiteraatiossa.

Dynaaminen analyysi suoritetaan tarkastelemalla eri interpolointimenetelmien käyttämää aikaa ennaltamäärätyn poikkileikkauksen muodostamiseen sekä arvioimalla näiden menetelmien tuottamien poikkileikkauksien visuaalista laatua. Yhtenä mittarina dynaamisessa analyysissä on siis suorituskyky, ja toisena visuaalisen tuotos. Suorituskyvyn vertailun toteutuksessa on kaksi olennaista yksityiskohtaa. Ensimmäiseksi syötteenä käytetty data on oltava samaa jokaisella keskenään verratulla testauskerralla. Toiseksi testauksessa käytetty laite ja sen tila on oltava vakio testikertojen välissä tarkkojen mittaustulosten saamiseksi. Esimerkiksi ylimääräiset ohjelmat taustalla voivat aiheuttaa väärentymiä suorituskyvyn mittaustuloksissa. Testitapausten yhtenäistämisellä pyritään eliminoimaan muiden tekijöiden kuin käytetyn interpolointimenetelmän, vaikutusta suorituskykyyn.

Kuten heuristisen arvioinnin tapauksessa, myös dynaamisen analyysin kohdalla ohjelmistomoduulista lähetetään testattava versio GTK:lle. Heiltä saatu palaute ohjaa dynaamisen analyysin ohella ohjelmistomoduulin kehittämistä, ennen kuin ohjelmistomoduulia lähdetään integroimaan GeoLab-ohjelmistoon.

4 Ohjelmistomoduulin kehittäminen

Tässä luvussa käydään läpi ohjelmistomoduulin suunnittelua, sen kehitystä sekä integraatiota GeoLab-ohjelmistoon. Moduulin kehitys on jaettu alalukuihin, joissa jokaisessa keskitytään moduulin suunnitteluun, integrointiin tai yksittäisen ominaisuuden kehitykseen. Luvusta 5 löytyy vastaavasti jokaista kehitettyä ominaisuutta ja integrointia käsittelevä alaluku, jossa käsitellään kehitettyjen ominaisuuksien tai integroinnin arviointia.

4.1 Ohjelmistomoduulin suunnittelu

Iivanainen ym. (2021a) kertovat GeoLabin olevan kehitetty pääosin C# ohjelmointikielellä. Tämän lisäksi GeoLabin käyttöliittymä on toteutettu .NET ohjelmistokehyksen tarjoamalla WPF-käyttöliittymäkirjastolla. Iivanainen ym. (2021a) kertovat toteuttaneensa useita käyttöliittymän ominaisuuksia onnistuneesti WPF-kirjaston käyttäjäkomponenttien[1] (eng. UserControl) avulla. Käyttäjäkomponentit ovat helppo keino luoda uusia muokattavia kokonaisuuksia käyttöliittymään perimällä WPF-kirjaston käyttäjäkomponentti luokan, jonka pohjalta voidaan luoda uusi kokonaisuus yhdistelemällä olemassa olevia käyttöliittymäkomponentteja. Uuden erillisen ominaisuuden toteuttaminen käyttäjäkomponenttina, tulee helpottamaan ominaisuuden integrointia GeoLabiin. Tämän lisäksi, koska käyttäjäkomponentit toteuttavat WPF:n käyttöliittymäkomponentti luokan, käyttäjäkomponenteista löytyy jo valmiiksi joitakin yleisiä käyttöliittymän toimintoja, kuten komponentin piilottaminen ja koon säätäminen. Toteuttamalla tutkielmassa kehitettävän ohjelmistomoduuli käyttäjäkomponenttina, helpottaa siis myöhemmin moduulin integrointia GeoLab-ohjelmistoon.

Koska käyttäjäkomponentti on erillinen kokonaisuus, se tarvitsee rajapinnan, jonka kautta sen toimintaa ohjataan käyttöliittymässä. Yleinen tapa ohjata käyttäjäkomponenttien[2] toimintaa on muuttaa käyttäjäkomponenttien ominaisuuksia (eng. properties). Jotta volumetrisesta datasta voidaan muodostaa mielivaltaisia poikkileikkauksia, tarvitaan tieto siitä mitä tiedostotyyppiä käytetään volumetrisen datan esittämiseen. Alaluvussa 2.3 käytiin läpi Geo-

1. https://learn.microsoft.com/en-us/dotnet/api/system.windows.controls.usercontrol
2. https://learn.microsoft.com/en-us/dotnet/desktop/wpf/controls

Labin täysin tukemat tiedostomuodot, jotka olivat kuvasekvenssi ja raakadata. Kuvasekvenssin kuvat sisältävät tiedon volumetrisen datan yksittäisen mittaustuloksen bittisyvyydestä ja kahden akselin dimensioista. Tällöin kuvien määrästä voidaan päätellä kolmannen akselin dimensio. Kuvasekvenssien tapauksessa käyttäjäkomponentti ei siis tarvitse erikseen tietoa volumetrisen datan dimensioista ja bittisyvyydestä. Iivanainen ym. (2021a) kertovat GeoLabin määrittelevän kuvasekvenssin suodattamalla tiedostosijainnista kaikki tiedostot, joiden tiedostonimen alku vastaa käyttäjän määrittelemää nimisuodatinta. Kuvasekvenssin tapauksessa käyttäjäkomponentti tarvitsee siis tiedon kuvasekvenssin tiedostosijainnista sekä nimisuodattimen, jonka avulla kuvatiedostot suodatetaan tiedostosijainnista kuvasekvenssiin. Raakadatan tapauksessa käyttäjäkomponentille täytyy kuvasekvenssin tavoin välittää tiedostosijainti. Koska raakadata ei sisällä muuta tietoa kuin pakkaamattomat mittaustulokset, käyttäjäkomponentti tarvitsee lisäksi tiedot volumetrisen datan dimensioista, yksittäisen mittaustuloksen bittisyvyydestä sekä bittijärjestyksestä. Molemmissa tapauksissa moduuli tarvitsee myös tiedon siitä mikä on näytettävä harmaasävyalue.

Koska tutkimuksessa käsiteltävä volumetrinen data on kolmiulotteista, on hyödyllistä pystyä esittämään kolmiulotteista avaruutta poikkileikkauksen näyttämisen yhteydessä. Tällöin käyttäjälle tarjotaan selkeä visuaalinen vastike siitä mistä kohtaa volumetrista dataa hänen määrittelemä poikkileikkaus on otettu. Kolmiulotteisen grafiikan esittämiseen on muutamia lähestymistapoja. Yksi vaihtoehto on käyttää WPF-kirjastosta jo valmiiksi löytyvää Viewport3D nimistä käyttöliittymäkomponenttia. Viewport3D on käyttöliittymäkomponentti, joka tarjoa työkalut luoda ja esittää kolmiulotteisen näkymän (Microsoft 2022).

Toinen tapa toteuttaa kolmiulotteista grafiikkaa on käyttää yleisiä 3D-grafiikkarajapintoja, kuten Direct3D tai OpenGL. Näiden käyttämisen olennaisin etu verrattuna Viewport3D käyttämiseen on se, että kehitettävä moduulin on helppo integroida .NET ohjelmistokehyksen ulkopuolelle. Tämä johtuu siitä, että nämä grafiikka rajapinnat ovat laajasti tuettuja. Tällöin tutkielman tulokset ovat helpommin yleistettävissä WPF-käyttöliittymäkirjaston ulkopuolella. Jotta näiden grafiikkarajapintojen käyttö saadaan integroitua GeoLabin käyttöliittymään, joudutaan käyttämään kirjastoa, jolla rajapintaa voidaan käyttää WPF-ympäristössä GeoLabin käyttämällä .NET 6 ohjelmistokehyksellä. OpenTK[3] ja SharpGL tarjoavat molemmat

3. https://opentk.net/

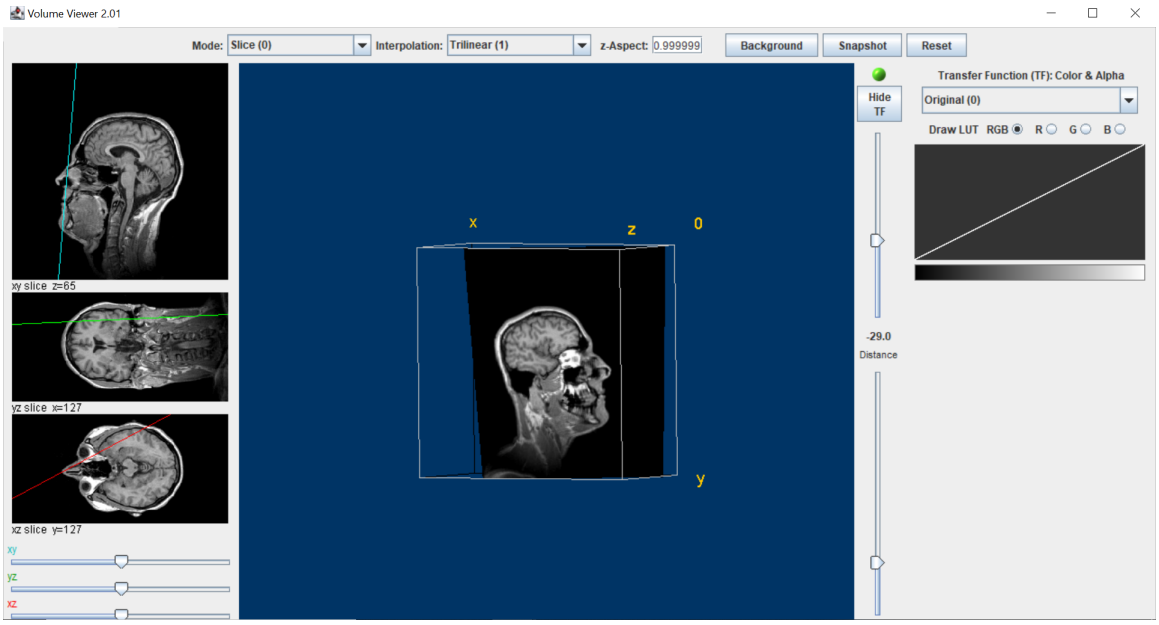
WPF:ään integroituvan OpenGL alustan. Jos näiden kahden kirjaston versiohallintoja[4] verrataan, voidaan todeta että OpenTK kehitys on aktiivisempaa. Sen takia tässä tutkimuksessa käytetään OpenTK:ta.

Koska kuvasekvensseissä volumetrisen datan mittaustulos esitetään kuvaformaateissa, kyky lukea alaluvussa 2.3 esitettyjä GeoLabin tukemia kuvatiedostoja on myös oleellinen ominaisuus, jonka toteuttaminen ei ole triviaalia. Myös kuvien näytettävän harmaasävyskaalan muuttaminen on oleellinen toiminto, jonka toteutettavan moduulin on tarjottava. Tämän takia tarvitaan keino lukea ja käsitellä GeoLabin tukemia kuvatiedostoja. Iivanainen ym. (2021a) kertovat, että GeoLabissa käytetään Magick.NET nimistä kuvankäsittelykirjastoa, jolla kuvasekvenssejä luetaan, ja jolla kuvien näytettäviä harmaasävyarvoja muokataan. Koska GeoLabista löytyy jo valmiiksi kuvankäsittelykirjasto, joka täyttää tutkielmassa kehitettävän ohjelmistomoduulin tarpeet, tätä kirjastoa voidaan käyttää myös toteutettavassa moduulissa.

4.2 Näkymän ja ohjauksen kehittäminen

Näkymän ja sen ohjauksen toteuttamisessa hyvä lähtökohta on tarkastella jo olemassa olevien volumetristä dataa käsittelevien ohjelmien yleisiä tai hyväksi todettuja ratkaisuja. Yksi yleisesti käytetty moniulotteisten kuvien kuvankäsittelyohjelma on alaluvussa 2.3 mainittu Fiji niminen avoimen lähdekoodin ohjelma. Schindelin ym. (2012) kertovat Fijin pohjautuvan hyvin paljon aikaisempaan ImageJ2 nimiseen ohjelmaan. Fiji tarkoituksena on laajentaa ImageJ2:sen toimintaa, mukana tulevien liitännäisten avulla. Schindelin ym. (2012) selittävät, että yksin Fijin vahvuuksista on ohjelmaa kehittävä, ylläpitävä ja käyttävä yhteisö. Schindelin ym. (2012) kertovat, että kuka tahansa voi ladata luomansa liitännäisen Fijin versiohallinnan kontribuutio haaraan, ja onnistuneen koodikatselmoinnin jälkeen liitännäinen ladataan Fijin päivitysten myötä käyttäjille saataviksi. Yksi Fijin liitännäisistä on Volume Viewer[5]. Kuvassa 6 esitetty Volume Viewer tarjoaa käyttäjilleen moniulotteisten kuvien visualisointia mielivaltaisten poikkileikkausten avulla. Volume Viewerin näkymässä näytetty data on Fijin mukana tuleva mallidatatiedosto nimeltä `T1-Head (16 bits)`.

4. https://github.com/opentk/opentk ja https://github.com/dwmkerr/sharpgl
5. https://imagej.net/plugins/volume-viewer



Kuvio 6: Fijin Volume Viewer liitännäisen käyttöliittymä.

Kuvassa 6 esitetty Volume Viewer tarjoaa käyttäjilleen erillisen ikkunan, jossa olennaisin elementti on keskellä oleva näkymä. Tässä näkymässä on volumetrisen datan dimensioita esittävä rautalankamalli. Rautalankamallin sivujen yhteydessä lukee X, Y tai Z riippuen mikä pääakseli on kyseessä sekä 0 volumetrisen datan avaruuden pistettä $(0,0,0)$ esittävän kulman kohdalla. Käyttäjä voi hiirellä raahaamalla pyörittää mallia näkymässä vapaasti. Rautalankamallin sisällä näkymää kohtisuorassa näkyy poikkileikkaus volumetrisesta datasta. Kun käyttäjä pyörittää rautalankamallia, hän samalla vaihtaa poikkileikkauksen tarkastelukulmaa. Tämän lisäksi kuvan oikealta puolelta löytyy kaksi liukusäädintä. Toisesta kuvaa voidaan tarkentaa tai loitontaa, ja toisesta liukusäätimestä voidaan säätä sitä, kuinka syvältä käyttäjän tarkastelukulmasta poikkileikkaus näytetään. Liitännäinen tarjoaa aluksi poikkileikkausta käyttäjän näkökulmasta keskeltä volumetrista dataa.

Liitännäisen ikkunassa näkyy myös kolmen eri pääakselien suuntaista poikkileikkausta ja niiden yhteydessä on kolme liukusäädintä. Näiden avulla käyttäjä voi vaihtaa pääakselien suuntaisten poikkileikkausten syvyyttä. Pääakselien poikkileikkausten yhteydessä näkyy myös viiva. Tämä viiva kuvaa mistä kohti keskellä näkyvä poikkileikkaus leikkaa pääakselin suuntaisia poikkileikkauksia. Tämän lisäksi liitännäinen tarjoaa käyttäjälleen eri interpolointimenetelmiä pudotusvalikon avulla. Interpolointimenetelmiä käydään tarkemmin läpi alalu-

vuissa 2.2 ja 4.3. Liitännäisen ikkuna tarjoaa myös painikkeesta avautuvan valikon, jonka avulla voidaan vaihtaa mielivaltaisen poikkileikkauksen näkymän taustaväriä.

Volume Viewerin käyttöliittymä tarjoaa käyttäjälle myös käyrän, jota hiirellä manipuloimalla käyttäjä voi valita miten erilaisia harmaansävyjä esitetään poikkileikkauksessa. Volume Viewer tarjoaa käyttäjille myös muita volumetrisen datan visualisointikeinoja, esim. projektiot ja pintahahmonnuksen, poikkileikkausten esittämisen lisäksi. Visualisointikeinoja voi vaihtaa lennosta visualisointimenetelmää määrittävän pudotusvalikon avulla. Päänäkymän oikealla puolella on myös vihreä pallo, joka viestittää käyttäjälle, että ohjelma ei tällä hetkellä suorita laskutoimituksia taustalla. Tämä pallo muuttuu punaiseksi silloin, kun ohjelma suorittaa visualisointiin liittyviä laskutoimituksia taustalla.

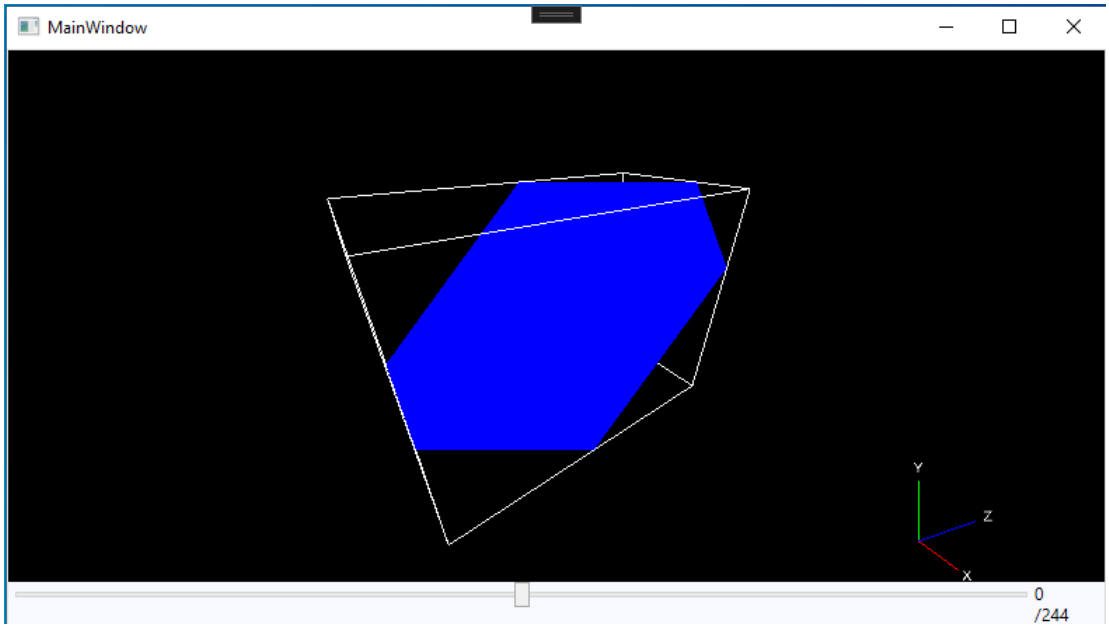
Kolmiulotteisen näkymän hallitseminen toteutetaan artefaktissa kaaripallokameran (eng. arcball camera) avulla. Shoemake (1994) kuvailee kaaripallokameran tarjoavan käyttäjälle nopeasti opittavan ja luontevan tavan tarkastella kolmiulotteista kohdetta eri näkökulmista. Shoemake (1994) kertoo, että kaaripallokameran helppokäyttöisyys perustuu siihen, että sen avulla kolmiulotteisen kappaleen tarkastelu toimii samankaltaisesti kuin oikeassa elämässä. Käyttäjä ottaa hiiren painiketta painamalla ja pitämällä pohjassa kiinni kappaleesta ja kääntelee sitä hiirtä liikuttamalla. Käyttäjä voi päästää irti kappaleesta lopettamalla hiiren painikkeen painamisen. Kaaripallokameran toiminta perustuu siihen, että käyttäjän hiiren liike muunnetaan kolmen pääakselin ympäri pyöriviksi rotaatioiksi. Esimerkiksi Fijin Volume Viewer liitännäinen käyttää kaaripallokameraa kolmiulotteisen näkymän hallitsemisessa. Kaaripallokameraan voidaan myös lisätä rajoitteita joidenkin akseleiden suhteen. Esimerkiksi Volume Viewerissä Z-akselin suuntainen rotaatio on rajoitettu kokonaan pois.

Tämän tutkielman kaaripallokameran toteutuksessa rajoitetaan Z-akselin rotaatio kokonaan pois, ja X-akselin liike on rajoitettu -90° ja 90° välille. Nämä rajoitteet perustuvat siihen, että tällöin kamera on aina orientoitunut niin, että kamerasta ylöspäin osoittava suunta osoittaa Y-akselin positiivista puolta päin. Tämä auttaa käyttäjää hahmottamaan, mistä suunnasta kappaletta tarkastellaan. Tämä ei myöskään rajoita käyttäjälle saatavilla olevia tarkastelukulmia, vaan käyttäjä voi rajoituksista huolimatta tarkastella kohdetta vapaasti kaikista mahdollisista kulmista.

Koska tutkielmassa keskitytään kehittämään viipalointia juuri suurikokoiselle volumetriselle datalle, on olennaista, että käyttäjä pystyy tarkentamaan, loitontamaan sekä liikuttamaan ohjelman näkymää halutessaan. Muuten suurikokoisissa tiedostoissa, kiinnostavien kohtien yksityiskohtainen tarkastelu olisi hyvin vaikeata. Iivanainen ym. (2021b) kertovat, että GeoLab-ohjelmalla voidaan tarkentaa ja loitontaa pääakselien suuntaisia poikkileikkauksia sekä liikuttamaan tarkasteltavaa aluettaan. Käyttäjä voi muuttaa poikkileikkauksen tarkennusta GeoLabissä hiiren rullan avulla pitämällä samanaikaisesti kontrollipainiketta pohjassa. Tämän takia on luontevaa lisätä myös toteutetussa moduulissa näkymän tarkennus ja loitonnus hiiren rullaan ja kontrollipainikkeeseen. GeoLabissa voidaan myös muuttaa tarkasteltavaa kohtaa panoroimalla (eng. pan) poikkileikkausta hiiren vasemmalla painikkeella. Hiiren vasenta painiketta käytetään jo moduulissa kuvakulman muuttamiseen, kuten Volume Viewerissä. Volume Viewer ei tarjoa vaihtoehtoa panoroida näkymää, joten moduulissa panorointitoiminto voidaan liittää hiiren oikeaan painikkeeseen. Tällöin käyttäjä voi hallita näkymää pelkän hiiren avulla.

Toinen olennainen osa näkymää ja sen ohjausta on poikkileikkauksen määrittäminen. Poikkileikkauksen määrittelyssä on kaksi olennaista asiaa. Kulma mistä poikkileikkaus otetaan, sekä miten poikkileikkauksen syvyys määritellään. Volume Viewer hoitaa poikkileikkauksen kulman määrittelyn niin, että poikkileikkaus on aina kohtisuorassa kameraa, jolloin kameran tarkastelukulman muuttaminen muuttaa myös samalla poikkileikkauksen kulmaa. Tätä tapaa voidaan käyttää myös toteutettavassa ohjelmistomoduulissa, koska kaksiulotteisia poikkileikkauksia on helpointa tarkastella juuri kohtisuorasta kulmasta. Näin myös näkymän ohjauksen hallitseminen pysyy hyvin yksinkertaisena, tehden moduulista käyttäjäystävällisen. Koska volumetrisen datan interpolointi on tarkoitus toteuttaa vasta myöhemmin, kuvasta 7 näkyy, että ohjelmistomoduulissa poikkileikkausta esitetään väliaikaisesti sinisenä alueena.

Poikkileikkauksen syvyyttä hallitaan niin GeoLabissa kuin Volume viewerissä liukusäätimen avulla, joten sen toteuttaminen moduulissa on intuitiivista toteuttaa myös liukusäätimellä. GeoLab tarjoaa myös pääakselien suuntaisten poikkileikkausten syvyyden säätämistä hiiren rullalla. Jos shift- tai kontrollipainiketta ei paineta samaan aikaan, kun hiirtä rullataan, syvyyttä muutetaan yhden yksikön verran. Jos taas hiirtä rullataan shift -painike on pohjassa, syvyyttä muutetaan 10 yksikön verran. Täten tämä ominaisuus on hyvä tuoda ohjelmistomo-



Kuvio 7: Ojelmistomoduulin näkymän ensimäinen iteraatio.

duuliin, jotta toiminta moduulin ja GeoLabin välillä pysyy yhtenäisenä.

Syvyyden liukusäätimen toteuttamisen yhteydessä tuli myös ilmi, että olisi hyvä, jos liukusäätimen mahdollisten arvojen joukko muuttuisi katselukulman vaihdoksen yhteydessä. Volume Viewer ei ota tarkastelukulmaa huomioon, vaan tarjoaa samat syvyyden arvojen joukot kaikille tarkastelukulmille. Volume Viewerissä tämä syvyyden arvoväli saadaan volumetristä dataa kuvaavan suorakulmaisen särmiön avaruuslävistäjän pituudesta. Koska keskipiste on tässä tapauksessa nolla, syvyyden arvoväli saadaan siis seuraavasti, jossa avaruuslävistäjän pituus on d.

$$n = d/2$$

Tällöin Volume Viewerin syvyyden arvoväli on $\{-n, \ldots, n\}$. Tässä yksinkertaisessa staattisessa tavassa tulee vastaan kuitenkin ongelmia, jos volumetrisen datan dimensioiden välillä on suuria eroja. Tällöin joidenkin tarkastelukulmien tapauksessa vain pieni osa syvyyden arvovälistä sisältää poikkileikkauksia, joiden taso sijoittuu edes osin volumetrisen dataalueen sisälle. Toisin sanoen, vain harvat syvyyden arvovälin arvot tuottavat näkyvän poikkileikkauksen volumetrisesta datasta, hankaloittaen liukusäätimen käyttöä. Ratkaisuna tähän

liukusäätimen arvoväliä voidaan säätää tarkastelukulman muutoksen yhteydessä, kattamaan vain näkyvät poikkileikkaukset. Koska jokaisesta tarkastelukulmasta viimeinen näkyvä poikkileikkaus on sellainen, joka koskettaa jotakin volumetrisen datan aluetta kuvaavan suorakulmaisen särmiön kulmaa, voidaan käyttää vektorien pistetuloa selvittämään, millä etäisyydellä viimeinen näkyvä poikkileikkaus on keskipisteestä. Merkitään kameran sijaintia näyttävän paikkavektorin yksikkövektoria $\vec{a}$. Koska poikkileikkaukset ovat kohtisuorassa kameraan, tiedetään että viimeisen näkyvän poikkileikkauksen keskipiste on $n\vec{a}$. Tämän lisäksi tiedetään, että poikkileikkauksen keskipiste on kohtisuorassa kulmassa jonkin volumetristä dataa kuvaavan alueen kulman kanssa. Jos volumetrisen datan leveyttä, korkeutta ja syvyyttä merkitään L, K ja S, niin vektori $\vec{b}$, joka kulkee viimeisen näkyvän poikkileikkauksen suuntaisesti poikkileikkauksen keskipisteen, ja volumetrisen data-alueen oikean ylätakakulman välillä voidaan muodostaa seuraavasti:

$$\vec{b} = (n\vec{a}) - (L/2, K/2, S/2)$$

Yllä oleva pätee vain niille kuvakulmille, joissa kulma, jonka kanssa viimeinen näkyvä poikkileikkaus on kosketuksissa, on volumetrisen data-alueen oikea ylätakakulma. Ratkaisu on tosin mahdollista yleistää kaikille tarkastelukulmille, joka tehdään myöhemmin. Koska n valitaan niin, että vektorit $n\vec{a}$ ja $\vec{b}$ ovat kohtisuorassa, vektorien pistetulo on tällöin 0, eli $(n\vec{a}) \cdot \vec{b} = 0$. Tällöin myös $\vec{a} \cdot \vec{b} = 0$, koska $\vec{a}$ on samansuuntainen $n\vec{a}$ kanssa. Tämän avulla voidaan ratkaista n, joka kertoo kuinka paljon poikkileikkausta voidaan siirtää keskipisteestä niin, että poikkileikkaus osuu volumetristä dataa kuvaavan suorakulmaisen särmiön kulmaan seuraavasti:

$$0 = \vec{a} \cdot \vec{b}$$
$$0 = (na_x - L/2)a_x + (na_y - K/2)a_y + (na_z - S/2)a_z$$
$$0 = na_x a_x - L/2a_x + na_y a_y - K/2a_y + na_z a_z - S/2a_z$$
$$n({a_x}^2 + {a_y}^2 + {a_z}^2) = L/2a_x + K/2a_y + S/2a_z$$
$$n = \frac{L/2a_x + K/2a_y + S/2a_z}{{a_x}^2 + {a_y}^2 + {a_z}^2}$$

Kuten aiemmin mainittiin, yllä oleva kaava pätee vain niille kaaripallokameran sijainneille, jotka sijaitsevat kaaripallon muodostaman pallon oikealla ylätakasektorilla. Käyttämällä hyväksi sitä, että volumetrinen data-alue on symmetrinen, voidaan kuitenkin todeta, että n on sama vaikka $\vec{a}$ peilattaisiin yhden tai useamman pääakselin suhteen. Tällöin peilaamalla kameran paikkavektori $\vec{a}$ itseisarvon avulla niin, että se sijaitsee volumetrisen data-alueen oikean ylätakakulman sektorilla, voidaan yleistää yllä olevan kaava kattamaan kaikki tarkastelukulmat seuraavasti:

$$n = \frac{L/2|a_x| + K/2|a_y| + S/2|a_z|}{|a_x|^2 + |a_y|^2 + |a_z|^2}$$

Tämän kaavan avulla voidaan siis laskea dynaamisesti, kuinka paljon poikkileikkausta voidaan siirtää kameraa kohti tai siitä poispäin, jotta poikkileikkaus on vieläkin näkyvissä. Tällöin syvyyden liukusäätimen minimi- ja maksimiarvot voidaan päivittää välille $\{-n, \ldots, n\}$ dynaamisesti kuvakulman muuttamisen yhteydessä. Tällöin käyttäjälle mielenkiintoinen, eli näkyvien poikkileikkausten alue kattaa koko liukusäätimen, tehden liukusäätimen käytöstä käyttäjälle helpompaa.

Ohjelmistomoduulin kehittämisen yhteydessä tuli ilmi, että datan orientaatio pitäisi esittää käyttäjälle. Kuten kuviosta 6 nähdään, Volume Viewerissä tämä hoidetaan niin, että volumetrisen datan piste $(0,0,0)$ merkitään käyttäjälle siten, että tämän kulman vieressä on numero 0. Nollakohdan vastaisissa kulmissa taas on merkitty joko X, Y tai Z, riippuen minkä pääakselin suuntaisesta vastakkaisesta kulmasta on kyse.

Tutkielman toteutuksessa, orientaatio päätettiin näyttää käyttäjälle käyttöliittymäelementtinä näkymän päällä. Tämä elementti näkyy käyttöliittymän kuvan 7 oikeassa alareunassa. Tämä elementti koostuu kolmesta erivärisestä viivasta, jotka muodostavat kulman. Tämä kulma vastaa Volume Viewerin kulmaa 0, eli tämä esittää kolmiulotteisen datan koordinaattia $(0,0,0)$. Viivojen värit vastaavat jokaista pääakselia vastaavaa värikoodausta, jotka voidaan tuoda parametrina ohjelmistomoduuliin. Tämän lisäksi viivojen päädyssä lukee, minkä pääakselin suuntainen viiva on kyseessä. Tämä viivojen muodostama kulma kääntyy kameran liikkeiden mukaan niin, että akselien viivat näyttävät aina missä suunnassa pääakselit ovat sillä hetkellä kameran näkökulmasta.

Näkymän ja ohjauksen ensimmäisessä arviointi-iteraatiossa löydettiin yhteensä viisi ongelmaa käytettävyyden suhteen. Arvioinnin pohjalta GTK:n kanssa tehtiin päätös, että näkymän ja ohjauksen käytettävyys ovat tarpeeksi hyvällä tasolla, jotta tutkimuksessa voidaan jatkaa interpoloinnin toteuttamiseen, kunhan löydetyt käytettävyysongelmat ovat korjattu. Arviointi-iteraation tarkempia tietoja voi lukea alaluvusta 5.1. Nämä viisi löydettyä käytettävyysongelmaa ovat tarkennustason näyttämisen puute, kaaripallokameran rajoitteet, panorointialueen liian pieni koko, pääakselin suunnat näyttävän käyttöliittymäkomponentin vaikealukuisuus, sekä se, että panorointi ei seuraa näkymässä hiiren paikkaa.

Ensimmäiseksi korjataan tarkennustason näyttäminen. OpenTK:n WPF-käyttöliittymäkomponentin päälle vasempaan yläreunaan laitettiin WPF:stä valmiiksi löytyvä textblock käyttöliittymäkomponentti, jonka teksti laitettiin muuttumaan tarkennustason muuttamisen yhteydessä. Seuraavaksi korjataan kaaripallokameran rajoitteet. Kuvakulman pyörittämisestä otetaan X- ja Z-akselin suuntaiset rajoitteet pois, jolloin käyttäjä voi täysin vapaasti pyörittää kaaripallokameraa haluamaansa suuntaan.

Panorointi on toteutettu niin, että kameran kiintopistettä sekä kameraa pystytään liikuttamaan kohtisuorassa olevan tason suuntaisesti. Tämä taso oli rajattu niin, että kiintopiste ei voi poistua volumetrisen datan dimensioita esittävän suorakulmaisen särmiön sisältä. Tämä kuitenkin aiheutti ongelman, että välillä käyttäjä ei voinut panoroida kuvaa tarpeeksi haluamiinsa suuntiin, joten panorointi rajattiin suuremmalle alueelle. Tämä alue on kuutio, jonka keskipiste on OpenGL avaruuden piste $(0, 0, 0)$ ja kuutio on niin iso, että käyttäjä voi tarkastelukulmasta huolimatta panoroida volumetrisen datan täysin pois näkymästä oletus tarkennustasolla.

Pääakselin suunnat näyttävä käyttöliittymäkomponentti poistettiin kokonaan. Sen sijaan volumetrisen datan dimensioita esittävän suorakulmaisen särmiön kulmaa $(0, 0, 0)$ ja sitä vastaavien pääakselin kulmien välille piirretyt janat värjättiin akseleille ominaisen värin mukaan, jotka tulevat GeoLabista. Koska GeoLabissa Y-akselin suuntaista kuvapinoa luetaan ylhäältä alas, eli toiseen suuntaan mitä OpenGL koordinaatisto kasvaa, tämän takia volumetristä dataa esittävän suorakulmaisen särmiön $(0, 0, 0)$ piste on särmiön vasemmassa etuyläkulmassa. Tämän yhteydessä tuli myös ilmi, että perspektiivinen projektio ei välttämättä ole otollinen tapa esittää volumetristä dataa. Tämän takia ohjelmistomoduulin lisättiin myös

vaihtoehto vaihtaa projektiota ortografiseen projektioon perspektiivisestä projektiosta. Tämä lisäksi poikkileikkausta esittävän alueen väri vaihdettiin sinisestä harmaaksi. Tämä tehtiin, koska harmaa kuvaa paremmin yksiväristen poikkileikkausten värimaailmaa, jolloin myös Y-akselin suuntainen sininen jana on helpompi erottaa ohjelmistomoduulin näkymästä.

Panoroinnin hiiren seurantaa pyrittiin muuttamaan niin, että hiiri pysyy koko ajan panoroitavan avaruuden pisteen päällä. Ortografisessa projektiossa tämä on yksinkertaista, koska näkymän näkymäpyramidi on suorakulmainen särmiö. Tällöin siis renderöityjen objektien koko pysyy samana niiden etäisyydestä kameraan huolimatta. Tällöin voidaan suoraan laskea, kuinka suuren matkan hiiri on kulkenut renderöidyssä näkymässä. Tämä onnistuu muuttamalla hiiren ruudulla kulkeman matkan näkymän koordinaateiksi. Koska ortografisesta projektiosta saadaan selville kuinka suuri näkymäpyramidi on näkymässä, voidaan hiiren kulkeman matkan muutos muuttaa panoroitavaksi suuruudeksi X- ja Y-komponenteille. Olettaen että ΔH on hiiren paikan muutos, näkymän leveys on L_n, näkymäpyramidin leveys on L_p, näkymän korkeus on K_n ja näkymäpyramidin korkeus on K_p. Voidaan X- ja Y-komponenttien panoroitavat suuruudet laskea erilisinä komponentteina seuraavasti:

$$\text{Pan}_x = \frac{\Delta H_x}{L_n} L_p$$
$$\text{Pan}_y = \frac{\Delta H_y}{K_n} K_p$$

Koska kameran yhteydessä pidetään tieto siitä mitkä ovat kamerasta suoraan ylöspäin ja oikealle osoittavat yksikkövektorit, voidaan kertoa ylöspäin osoittava vektori Pan_y:llä ja oikealle päin osoittava vektori Pan_x:llä. Tällöin saadaan kaksi vektoria, joiden suuntaisesti pitää liikuttaa näkymää, jotta panorointi seuraisi hiirtä ortografisessa projektiossa. Perspektiivisessä projektiossa näkymän panoroinnissa on otettava huomioon perspektiivi, eli se, että renderöitävät objektit pienenevät sitä mukaan, kun ne menevät kauemmaksi, sekä suurenevat sitä mukaan, kun ne tulevat lähemmäksi. Tällöin panorointiin vaikuttaa myös se, mikä on panoroinnin kiintopiste, eli etäisyys mistä kohtaa näkymää tahdotaan panoroida. Hyvä vaihtoehto panoroitavaksi etäisyydeksi on valita kameran etäisyys poikkileikkauksesta, koska näkymän ensisijainen funktio on visualisoida volumetristä dataa poikkileikkausten avulla. Oletetaan, että näkökenttä (eng. field of view) on N, kameran etäisyys panoroitavasta syvyydestä on

E_k, ja kuvasuhde on R. Tällöin panoroitavat X ja Y komponentit voidaan laskea erillisinä komponentteina seuraavasti:

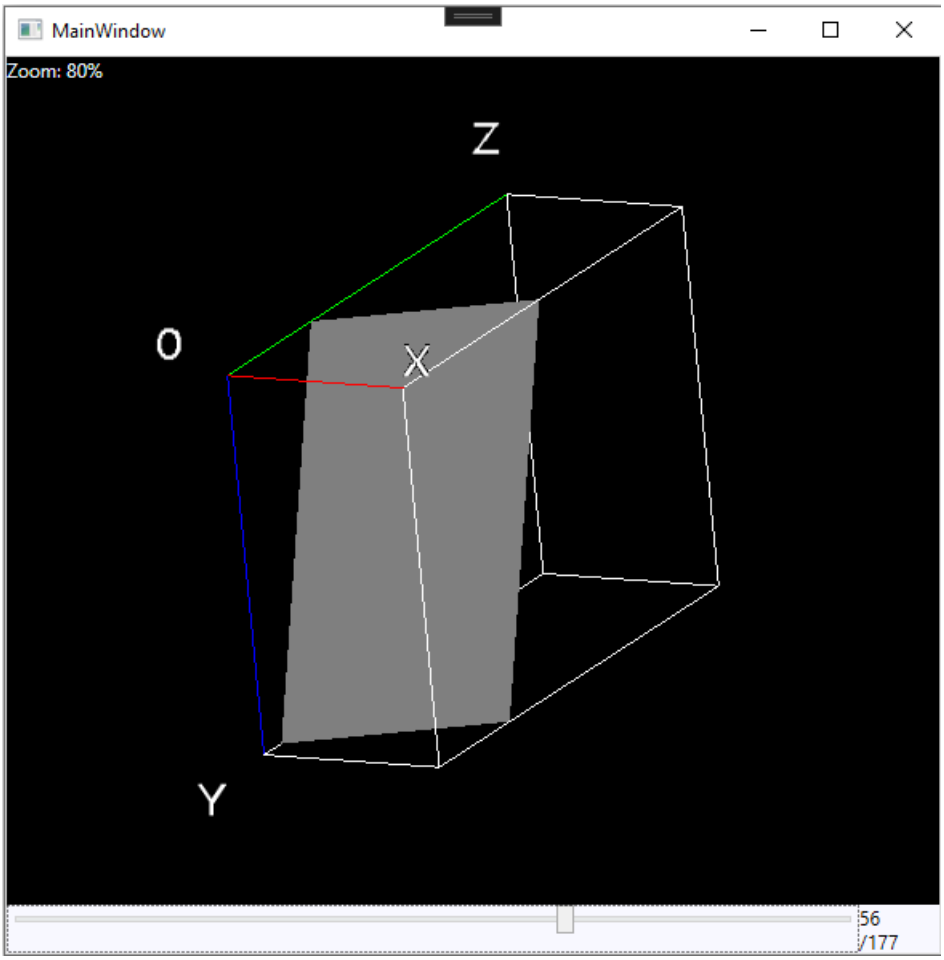
$$\text{Pan}_x = \frac{\Delta H_x}{L_n} 2Tan(N/2)E_k R$$
$$\text{Pan}_y = \frac{\Delta H_y}{K_n} 2Tan(N/2)E_k$$

Tämän jälkeen voidaan liikuttaa Pan_x ja Pan_y avulla näkymää kameran suuntavektorien osoittamiin suuntiin samalla tavalla kuin ortografisessa projektiossa. Korjausten jälkeinen versio ohjelmistomoduulin näkymästä esitetään kuvassa 8. Kuvasta voidaan nähdä ortografinen projektio, tarkennustason näyttäminen sekä uusi tapa näyttää käyttäjälle näkymän orientaatio avaruuden suhteen.

4.3 Volumetrisen datan interpoloinnin kehittäminen

Kun käyttöliittymä ja sen ohjaus on saatu käyttökelpoisiksi, voidaan toteutuksessa siirtyä interpolointiin. Jotta volumetrisesta datasta voidaan visualisoida mielivaltaisia poikkileikkauksia, on yleistä, että poikkileikkauksen volumetrisen datan arvoja esittävät pikselit eivät ole suoraan minkään volumetrisen datan mittaustuloksen arvon $S(x,y,z)$ kohdalla. Tällöin arvo joudutaan arvioimaan jollain menetelmällä, ja näitä menetelmiä kutsutaan yleisesti interpolointimenetelmiksi. Alaluvussa 2.2 käytiin läpi kuusi yleistä interpolointimenetelmää kolmiulotteiselle datalle. Nämä menetelmät ovat lähimmän naapurin interpolointi, luonnollisen naapurin interpolointi, kolmilineaarinen interpolointi, kolmikuutio interpolointi, B-mutka interpolointi sekä kriging interpolointi. Vaikka interpolointimenetelmiä on paljon, näiden menetelmien joukosta lähimmän naapurin interpolointi, kolmilineaarinen interpolointi sekä kolmikuutio interpolointi ovat yleisimmin käytettyjä isotrooppisten volumetristen datojen interpoloinnissa. Tämä johtuu siitä, että nämä menetelmät tarjoavat tarpeeksi tehokasta suorituskykyä ja hyvää kuvanlaatua, jotta niitä voitaisiin käyttää suurikokoisen volumetrisen datan interpolointiin reaaliajassa.

Hyvä lähtökohta interpolointimenetelmän toteuttamiselle on tarkastella aikaisempien tutkimusten tuloksia ja mitä jo olemassa olevat ohjelmat tarjoavat interpolointimenetelmien vaih-



Kuvio 8: Ohjelmistomoduulin näkymä ensimmäisen arvioinnin korjausten jälkeen.

toehdoiksi. Csébfalvi (2019) kuvailee kolmilineaarisen interpoloinnin olevan standardimenetelmä volumetrisen datan interpolointiin. Bourke (1999) kertoo myös, että kolmilineaarisen interpoloinnin mahdollisesti yleisin käyttötarkoitus on volumetrisen datajoukon mittausarvojen $S(x,y,z)$ diskreettien pisteiden välisten arvojen arviointi. Kolmilineaarinen interpolointimenetelmä on myös yksi neljästä interpolointimenetelmästä, joita Fijin Volume Viewer liitännäinen tarjoaa. Tämä tekeekin kolmilineaarisesta interpoloinnista otollisen vaihtoehdon lähteä toteuttamaan interpolointia.

Toisaalta luvussa 3.6 käydään läpi kuinka interpolointimenetelmiä on tarkoitus arvioida vertaamalla niiden suorituskykyä sekä tuotetun kuvan laatua. Tämä tarkoittaa sitä, että tutkimuksessa toteutetaan ainakin kaksi interpolointimenetelmää, jotta niitä voidaan analyyttisesti arvioida keskenään. Koska Kaufman (2000) kuvailee lähimmän naapurin interpoloin-

nin olevan yksinkertaisin interpolointimenetelmä, sen toteuttaminen ensiksi antaa hyvän vertailukohteen menetelmien väliselle arvioinnille.

On otettava huomioon, että toteutuksessa on tarkoitus esittää poikkileikkauksia suurikokoisesta volumetrisesta datasta. Tämä tarkoittaa sitä, että data ei todennäköisesti mahdu kokonaisena koneen keskusmuistiin. Esimerkiksi Fiji:ssä oletuksena koko volumetrinen data ladataan sovellusvälimuistiin ennen kuin sitä voidaan käsitellä. Fiji myös tarjoaa isoille datoille ratkaisuksi liitännäisen kautta virtuaalipinon (eng. virtual stack), jonka avulla volumetrista dataa luetaan dynaamisesti massamuistista sitä mukaan, kun sitä tarvitaan. Virtuaalipinojen avulla voidaan tosin lukea poikkileikkauksia vain volumetrisen datan esityssuunnassa.

Tutkimuksessa kehitettävän ohjelmistomoduulin on siis myös tarve lukea volumetrista dataa sitä mukaan, kun arvoja tarvitaan visualisointiin. Tämä aiheuttaa pullonkaulan suorituskyvyssä, koska massamuistista lukeminen on huomattavasti hitaampaa kuin vastaavan datan lukeminen keskusmuistista. Tämän lisäksi luvussa 2.4 käytiin läpi, että käytettävät tiedostoformaatit eivät tarjoa sitä, että luettavat arvot olisivat jotenkin etukäteen jaoteltu. Tällöin vierekkäiset mittaustulokset eivät siis luultavasti ole toistensa lähettyvillä tiedostossa. Esimerkiksi kuvasekvenssin tapauksessa poikkileikkauksen vierekkäiset mittaustulokset voivat olla eri tiedostoissa.

Koska massamuistista luettavat pisteiden arvot eivät todennäköisesti sijaitse vierekkäin, on massamuistista tiedoston osien lukeminen suoritettava usealla eri tiedostonlukuoperaatiolla. Pahimmassa tapauksessa jokaiseen pisteen interpolointiin on luettava massamuistista arvo erillisellä tiedostonlukuoperaatiolla. Tämä pahentaa suorituskyvyn pullonkaulaa, koska tiedoston lukeminen tavu kerrallaan on huomattavasti hitaampaa kuin se että koko tiedosto luettaisiin yhdellä tiedostonlukuoperaatiolla. Tämän vaikutus suorituskykyyn on vielä suurempi, jos käytetään interpolointimenetelmää, joka tarvitsee usean mittaustuloksen arvon interpoloitavan pisteen läheltä. Tällöin jokaista interpoloitavaa arvoa varten, on mahdollisesti tehtävä useampi tiedostonlukuoperaatio. Näissä tapauksissa on kuitenkin yleistä, että interpolointiin käytettävistä arvoista ainakin osat arvoista sijaitsevat vierekkäin tiedostossa, jolloin on mahdollista lukea useita interpolointiin käytettäviä arvoja yhdellä tiedostonlukuoperaatiolla.

Toinen olennainen yksityiskohta interpoloinnin toteutuksessa on se, millä välein volumet-

rista dataa interpoloidaan. Yksi vaihtoehto interpoloinnin väleille on interpoloida volumetristä dataa samassa suhteessa datan mittaustulosten välien kanssa. Tämä tarkoittaa siis sitä, että kahden datan mittaustuloksen väli on sama kuin mitä kahden interpoloitavan pisteen välillä. Tällä interpolointivälillä volumetrisen datan pääakselien suuntaisten poikkileikkauksien pisteet sijoittuvatkin täysin volumetrisen datan mittaustulosten arvojen kohdalle, jolloin varsinaista interpolointia ei tarvita. Kuten luvussa 2.2 kerrotaan, interpolointifunktio vastaa mittaustuloksen arvoja silloin, kun interpoloitava piste on täysin mittaustuloksen kohdalla. Tällöin $f(x,y,z) = S(x,y,z)$, jossa f on interpolointifunktio pisteelle (x,y,z), ja $S(x,y,z)$ on volumetrisen datajoukon arvo diskreetissä pisteessä. Tällä menetelmällä poikkileikkauksen arvot joudutaan interpoloimaan uudestaan silloin, kun poikkileikkaus muuttuu. Toisin sanoen interpolointi joudutaan tekemään silloin, kun poikkileikkauksen tarkastelukulma tai syvyys muuttuu.

Toinen mahdollinen vaihtoehto poikkileikkauksen interpoloinnin pisteille on interpoloida poikkileikkausta näkymän (eng. viewport) pikselien kohdalta. Tällä menetelmällä, jokaiselle näkymän pikselille tehdään tarkistus osuuko pikseli poikkileikkaukseen. Jos osuu, interpoloidaan poikkileikkauksen arvo tästä osumakohdasta. Tämän menetelmän etuna on se, että se skaalaa poikkileikkauksen tarkkuuden näkymän suhteen. Esimerkiksi jos suurikokoisen datan poikkileikkausta katsotaan kaukaa, dataa interpoloidaan väljemmin, koska näkymä ei esittäisi interpoloidun datan kuvaa tarpeeksi tarkasti. Myös päinvastaisesti, jos näkymä on tarkennettu hyvin lähelle volumetrisen datan poikkileikkausta, arvoja voidaan interpoloida useita kahden mittaustuloksen välillä. Tässä tapauksessa on tosin huomioitava, että on käytettävä interpolointimenetelmää, joka ottaa huomioon sen, kuinka kaukana interpoloitava piste on mittaustuloksesta. Esimerkiksi lähimmän naapurin interpolointimenetelmä, ei tuottaisi tietyn tarkennustason jälkeen yhtään tarkempaa kuvaa. Tällä menetelmällä voidaan myös rajata interpoloitavaa aluetta, jolloin interpolointia ei tarvitse suorittaa niiltä osilta, mitkä eivät näy kuvassa.

Huonona puolena tässä menetelmässä on se, että interpolointia joudutaan suorittamaan näkymän jokaisen pikselin kohdalta, jolloin näkymän koko vaikuttaa interpoloitavien pisteiden määrään. Koska näkymän muutos vaikuttaa interpoloitaviin arvoihin, joudutaan arvot laskemaan uudelleen silloin, kun poikkileikkaus tai näkymä muuttuu. Poikkileikkauksen interpo-

loidut arvot joudutaan laskemaan uudestaan silloin, kun poikkileikkauksen tarkastelukulma tai syvyys muuttuvat. Tämän lisäksi arvoja joudutaan interpoloimaan uudestaan myös silloin, jos käyttäjä tarkentaa tai panoroi näkymää. Myös komponentin koon muuttaminen muuttaa näkymän kokoa, joten myös silloin interpoloitavat arvot joudutaan laskemaan uudestaan.

Näistä kahdesta eri interpoloinnin pisteiden välistä, tässä tutkielmassa pisteitä interpoloidaan samassa suhteessa lähtödatan kanssa. Vaikka tämän menetelmän interpolointi suurikokoisella datalla tulee luultavasti kestämään kauemmin kuin silloin jos interpolointia toteutetaan vain näkymän pikselien kohdalta, tässä menetelmässä arvoja ei tarvitse laskea uudestaan, kun näkymä muuttuu. Olettaen että käyttäjä tahtoo tarkastella yksittäistä poikkileikkausta tarkemmin, on suotavampaa, että interpolointi hoidetaan poikkileikkauksen osalta heti, jonka jälkeen käyttäjä on vapaa tarkastelemaan poikkileikkausta ilman uudelleen interpolointia.

Koska yksi tutkimuksessa kehitettävän ohjelman vaatimuksista on se, että se tarjoaa käyttäjälleen vastetta syötteeseen reaaliajassa. Tällöin ohjelma ei voi laskea kaikkia interpoloitavan pisteiden arvoja poikkileikkauksen muuttuessa, koska tämä pysäyttäisi ohjelman toiminnan laskutoimitusten ajaksi. Luebke ym. (2003) kuvailevat ohjelman monimutkaisuuden ja suorituskyvyn tasapainottelun olleen jo pitkään yleisesti ongelmana tietotekniikan alalla. He kuvailevat ratkaisuksi tähän ongelmaan graafisen ohjelmoinnin yhteydessä olevan tarkkuustasot (eng. level of detail). Tarkkuustasomenetelmiä yhdistää se, että niiden tarkoituksena on vähentää ongelmasta yksityiskohtia, jolloin ratkaisun suorituskyky paranee. Tällä tavoin voidaan tarjota kompromissi monimutkaisuuden ja suorituskyvyn välillä. Graafisen ohjelmoinnin yhteydessä monimutkaisuus tarkoittaa yleensä sitä, kuinka paljon piirrettäviä yksityiskohtia näkymä sisältää. Suorituskyvyllä taas tarkoitetaan kuinka nopeasti näkymän saadaan valmiiksi.

Soveltamalla tarkkuustasoja kehitettävässä ohjelmistomoduulissa, käyttäjälle voidaan tarjota pienen tarkkuustason kuva reaaliajassa, ja jatkaa taustalla kuvan tarkentamista ajan myötä. Tätä samaa periaatetta käytetään myös hyväksi Fijin Volume Viewer-liitännäisessä, ja alaluvussa 4.2 käydään läpi, kuinka Volume Viewer liitännäinen myös viestii käyttäjälleen siitä, että ohjelmisto toteuttaa visualisointiin liittyviä laskutoimituksia taustaoperaationa väriä muuttavan pallon avulla.

Luvussa 4.2 esitetty poikkileikkausta esittävä taso on toteutettu niin, että volumetrisen datan dimensioita kuvaavan särmiön Z-akselin nollakohtaan on piirretty neliö, joka on kooltaan tarpeeksi suuri peittämään kokonaisen tason särmiöstä kaikista tarkastelukulmista ja syvyyksistä. Poikkileikkauksen syvyyden muuttaminen siirtää tätä neliötä Z-akselin suuntaisesti eteen- tai taaksepäin. Tämän jälkeen neliön paikkaa muutetaan kameran rotaation mukaisesti, jotta poikkileikkaus on kohtisuorassa kameraan. Lopuksi varjostimen puolella poikkileikkauksesta jätetään volumetrisen data-alueen ulkopuoliset kohdat neliöstä piirtämättä. Näkymän kehitysvaiheessa tähän tasoon lisätään tekstuuri, joka luodaan volumetrisesta datasta kuvaamaan poikkileikkauksen varsinaisia interpoloituja arvoja.

Poikkileikkauksen esittäminen isona tasona, joka leikataan varjostimen puolella, tuottaa kuitenkin suorituskyvyllisiä ongelmia, jos poikkileikkauksen pisteiden arvoja lähdetään interpoloimaan suoraan tason pisteiden koordinaatteihin. Vaikka onkin huomattavasti nopeampaa tarkastaa se, että tason kohta ei ole volumetrisen datan sisällä, jolloin ohjelmassa voidaan välttyä tiedostonlukuoperaatiolla, taso sisältää silti huomattavan määrän pisteitä, joille tämä tarkistus joudutaan tekemään. Tutkimuksessa löydettiin ratkaisu tähän ongelmaan esim. `TIFF` ja JPG kuvatiedostoformaateista löytyvästä ominaisuudesta jakaa yksittäinen kuva erillisiin neliön muotoisiin osiin (eng. image tiling).

Skodras, Christopoulos ja Ebrahimi (2001) kertovat, että jakamalla kuvatiedoston osiin, on mahdollista lukea pienempiä kokonaisuuksia kuvasta, sekä mahdollisuus käsitellä jokaista osiota erikseen, täten vähentäen kuvan prosessointiin tarvittavaa muistia. Tässä tutkielmassa käytetään hyväksi kuvan jakamista osiin, jakamalla poikkileikkauksen määrittämä taso pienempiin neliönmuotoisiin osiin. Näiden osien avulla säästytään huomattavalta määrältä laskutoimituksia, koska ohjelmassa voidaan yhdellä laskutoimituksella tarkastaa, onko osa tarpeeksi lähellä volumetrisen datan aluetta, jotta se voisi sisältää pisteitä, jotka sijoittuvat volumetrisen datan alueelle. Tämä myös säästää jonkin verran muistia, koska ohjelman ei tarvitse pitää muistissa kuvan osia, jotka eivät sijoitu volumetrisen datan alueelle.

Edellä kuvattu ositus tarjoaa myös luontevan lähtökohdan toteuttaa taustaprosessina tarkentuvat tarkkuustasot. Ohjelma voi aluksi interpoloida vain yhden pisteen arvon osaa kohti, ja jatkaa taustalla osien tarkentamista. Myös toinen etu mikä tästä seuraa, on se, että eri osien tekstuurien data voidaan pitää erillisissä listoissa. C# käyttää oletuksena listojen indeksei-

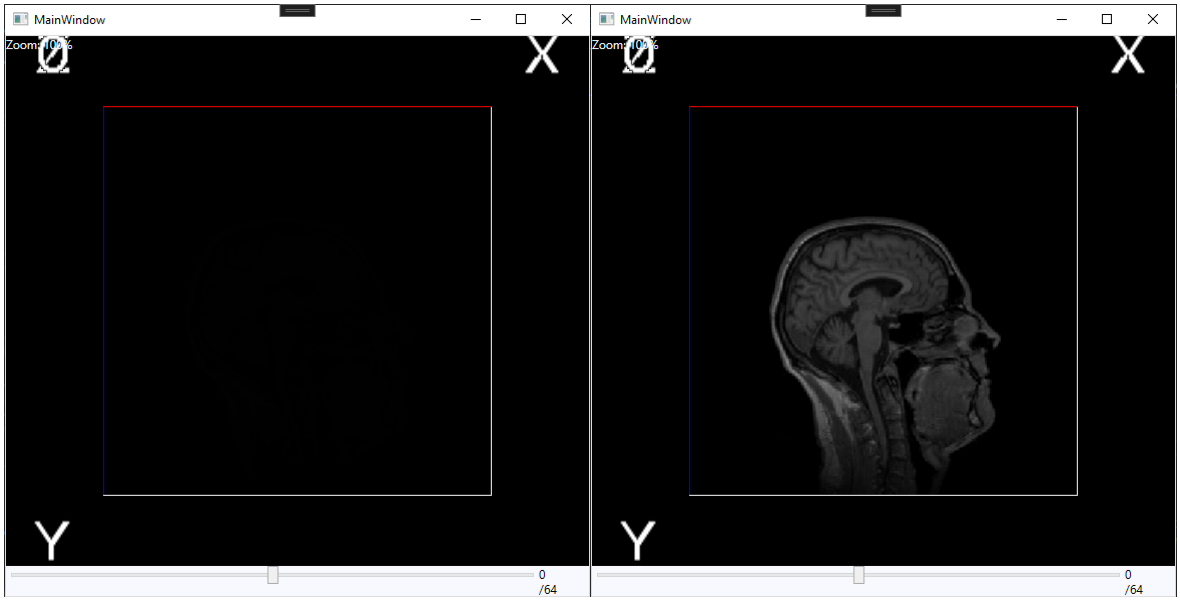
nä etumerkillisiä 32-bittisiä kokonaislukuja. Tämä tarkoittaa, että tavutaulukkoon mahtuu tasan kahden gigatavun verran erillisiä tavuja. Tämä voisi isojen datojen tapauksissa osoittautua riittämättömäksi. Menetelmä jossa pidetään poikkileikkauksen interpoloituja arvoja taulukossa tavutaulukkoja aiheuttaa tosin ongelmia renderöinnin puolella. Koska pikselidata ei ole yhden C# olion sisällä, ei ole mahdollista piirtää kaikkia poikkileikkauksen osia yhdellä piirtokomennolla, jolloin renderöinti on hieman hitaampaa. Tämä ei tosin muodosta ongelmaa, jos erikseen renderöitävien osien määrä pidetään suhteellisen matalana.

Kehitettävän ohjelmistomoduulin pääasiallinen tarkoitus on antaa käyttäjälleen työkalu volumetrisen datan visualisointiin. Tämän takia on tärkeää, että käyttäjä voi vaihtaa näytettävää harmaasävyskaalaa. Kuvassa 6 esitetty Fijin Volume Viewer liitännäinen toteuttaa tämän antamalla käyttäjän vapaasti muokata poikkileikkauksen värihistogrammia, joka löytyy liitännäisen oikeasta yläkulmasta. Fiji myös yrittää arvata käytettävän harmaasävyskaalan katsomalla mitä arvoja volumetrinen data sisältää. GeoLab, johon kehitettävä ohjelmistomoduuli on tarkoitus liittää tutkimuksessa, tarjoaa yksinkertaisemman tavan muokata näytettäviä harmaasävyarvoja.

GeoLabin oikeassa reunassa on kaksi liukusäädintä ja tekstinsyöttökenttää, joilla voidaan muuttaa näytettävien harmaasävyarvojen minimi- ja maksimiarvoja. Tämä tarkoittaa, että kaikki arvot minimiarvon alapuolella esitetään täysin mustana, ja kaikki arvot maksimiarvon yläpuolella esitetään täysin valkoisena. Näiden kahden arvon väliin jäävät arvot taas normalisoidaan kattamaan koko näytettävä harmaasävyskaalaa. Koska GeoLab ei automaattisesti lataa volumetrista dataa kokonaan muistiin, GeoLab ei myöskään anna käyttäjälle arvausta siitä mikä voisi olla näytettävä harmaasävyskaala, vaan tarjoaa käyttäjälleen oletuksena käytetyn formaatin minimi- ja maksimiarvoja. Näitä arvoja GeoLab käsittelee sisäisesti prosentteina, joten kehitettävän ohjelmistomoduulin on tarjottavan harmaasävyskaalan normalisointi kahden prosenttiarvon välille.

Näytettävän harmaasävyskaalan muuttaminen on tarpeellista silloin, kun käsiteltävän volumetrisen datan arvojoukko sijoittuu vain pienelle välille käytetyn volumetrisen datan formaatin mahdollisista arvoista. Esimerkkinä kuvassa 9 näytetään kuinka volumetrisen datan poikkileikkaus näyttää vasemmassa kuvassa täysin mustalta oletus harmaansävyskaalalla. Oikeanpuoleisesta kuvasta poikkileikkauksesta erottuu selvästi tietokonetomografialla ku-

vannettu pää. Käytettynä volumetrisena datana on Fijistä löytyvä mallitiedosto `T1-Head (16 bits)`, joka on 16-bittistä volumetrista dataa, jolloin jokainen datan pisteen arvo voi sijoittua välille $[0, 65535]$. Datassa kaikkien pisteiden arvot liikkuvat kuitenkin korkeintaan muutamissa tuhansissa, joten oletusskaalalla volumetrinen data ei käytännössä ole ollenkaan näkyvissä.



Kuvio 9: Volumetrisen datan näkymä kahdella eri harmaansävyskaalalla.

Harmaasävyskaalan normalisointi on laskennallisesti kevyt prosessi, joten sen voi hoitaa reaaliajassa näkymän renderöitiprosessin aikana pikselivarjostimen puolella. Tällöin volumetrisesta datasta luetut arvot voidaan säilyttää muistissa muuttumattomana, jolloin normalisointi ei kadottaisi yksityiskohtia datasta. Tämän avulla samalla poikkileikkauksen datalla voidaan suorittaa useita eri harmaasavyskaalan normalisointeja ilman, että pisteiden arvoja joudutaan interpoloimaan uudestaan.

Patro ja Sahu (2015) kertovat, että arvo voidaan normalisoida minimi- ja maksimiarvojen välille seuraavasti. Oletetaan, että v on jokin normalisoitava arvo. Tällöin v_{min} on normalisointivälin minimiarvo ja v_{max} on normalisointivälin maksimiarvo. Patro ja Sahu (2015) esittävät myös, että jos arvoa on normalisoitu jollekin välille aikaisemmin, niin tarvitaan kaksi muuttujaa lisää. Muuttuja A on arvon mahdollisen aiemman normalisoinnin maksimiarvo ja B on mahdollisen aiemman normalisoinnin minimiarvo. Koska kuvan varjostimen puolella

käytetään alkuperäistä mittausdataa, joka on skaalattu välille $[0,1]$, tarkoittaa se, että $A = 1$ ja $B = 0$. Tästä saadaan muodostettua normalisoidulle arvolle v_n kaava seuraavasti:

$$v_n = \frac{v - v_{min}}{v_{max} - v_{min}}(A - B) + B = \frac{v - v_{min}}{v_{max} - v_{min}}(1 - 0) + 0 = \frac{v - v_{min}}{v_{max} - v_{min}}$$

Luvussa 2.3 käytiin läpi eri formaatit joita GeoLab tukee täysin. Nämä formaatit ovat raakadata ja kuvasekvenssi. Raakadata sisältää vain mittaustulosten arvoja toisensa jälkeen. Raakadatan metatiedot, eli volumetrisen datan leveys, korkeus ja syvyys, datan arvojen käyttämä tavusyvyys ja tavujärjestys, sekä pääakseli, jonka suuntaisesti data on esitetty, saadaan GeoLabin puolelta. Näiden arvojen avulla on mahdollista lukea yksittäisen mittaustuloksen arvo pisteestä (x,y,z). Toteutuksessa on huomioitava, että koordinaatit ovat muutettava tiedoston lukemista varten luonnollisten lukujen määrittämiksi indekseiksi. Jos merkitään, että volumetrisen datan leveys on L, korkeus on K, syvyys on S ja tavusyvyys on T, mittaustulos $S(x,y,z)$ löytyy tiedostosta kohdasta $(x+\frac{L}{2}+(y+\frac{K}{2})L+(z+\frac{S}{2})KL)T$, olettaen että tiedosto esittää volumetrista dataa Z-akselin suuntaisesti. Yksittäisen mittaustuloksen lukemisen yhteydessä on myös mahdollista kääntää luetun mittaustuloksen tavut toisinpäin. Tämä on tarpeellista, jos datassa käytetään eri tavujärjestystä, mitä tutkimuksessa kehitettävä ohjelmistomoduuli käyttää.

Kuvasekvenssin tapauksessa, itse interpolointimenetelmät toimivat samalla tavalla kuin raakadatan tapauksessa, mutta mittaustuloksen $S(x,y,z)$ lukeminen eroaa huomattavasti raakadatasta. Luvussa 2.3 käytiin läpi GeoLabin tukemat kuvatiedostoformaatit, jotka ovat `PNG`- ja `TIFF`-tiedostot. Luvussa 2.3 käytiin läpi `TIFF`-tiedostojen joustava ominaisuus sisältää pakkaamatonta tai pakattua dataa. Pakkausmenetelmät, joita `TIFF`-tiedostot voivat hyödyntää, ovat moninaisia. Pakkaamattoman datan ja joidenkin pakkausalgoritmien tapauksissa yksittäisen mittaustuloksen $S(x,y,z)$ lukeminen olisi mahdollista toteuttaa laskemalla missä kohtaa tiedostoa haluttu mittaustulosta esittävä pikselin arvo on. Jotkin pakkausmenetelmät, kuten luvussa 2.3 mainittu `PNG`-tiedostojen käyttämä pakkausmenetelmä, eivät tosin mahdollista tätä. Tämä johtuu siitä, että pakatun pikselin arvon saamiseksi pahimmassa tapauksessa koko kuvatiedosto on käytävä läpi, koska pakkausalgoritmi poistaa toistuvia pikseliarvoja, jolloin pikselin arvo voi olla referenssi aiemman pikselin arvoon.

Koska GeoLab käyttää jo valmiiksi ulkoista Magick.NET kirjastoa kuvatiedoston lukemiseen, ja koska erillisen kuvatiedostojenluku kirjaston toteuttaminen on tämän tutkielman aiheen ja resurssien ulkopuolella, tyydytään tutkielmassa käyttämään Magick.NET kirjaston kuvien lukemiseen. Tämä tarkoittaa, että yksittäisen kuvan mittaustuloksen arvoa $S(x,y,z)$ esittävän pikselin lukemiseksi, tutkielmassa toteutettavan ohjelmistomoduulin on luettava kokonainen kuvasekvenssin poikkileikkaus, jotta yksittäisen pikselin arvo saataisiin selville. Tällöin kuvasekvenssien tapauksessa poikkileikkausten esittäminen on huomattavasti hitaampaa, koska jokaisen poikkileikkauksen pikselin arvoa luettaessa joudutaan massamuistista lukemaan vähintään yksi kuvasekvenssin kuva.

Kuvasekvenssin mittaustulosten lukemista on mahdollista silti optimoida hieman pitämällä luettua kuvaa muistissa. Tällöin jos seuraavan luettavan pikselin arvo löytyy jo sovellusvälimuistissa pidetystä kuvasekvenssin kuvasta, on mahdollista välttyä massamuistin lukuoperaatioiden suorittamiselta kokonaan pisteen arvon interpoloinnin yhteydessä. Koska tutkielmassa pyritään tekemään ratkaisu, jolla voidaan käsitellä suurikokoisia volumetrisiä datoja, datan mittasuhteista riippuen yksittäinenkin kuva voi olla kooltaan hyvin suuri. Tämän takia useita kuvatiedostoja ei voida pitää sovellusvälimuistissa. Tutkielmassa toteutettavan moduulin tapauksessa sovellusvälimuistissa pidettävien kuvien määrä on konservatiivinen. Tämä tarkoittaa, että kuvatiedostoja pidetään sovellusvälimuistissa korkeintaan niin monta kuin yhden mittaustuloksen arvon $S(x,y,z)$ interpolointiin tarvitaan, eli tämä riippuu interpolointimenetelmästä. Esimerkiksi lähimmän naapurin interpoloinnissa sovellusvälimuistissa pidetään korkeintaan yhtä kuvaa kerrallaan.

Aiemmin mainitussa tarkkuustason toteutuksessa voidaan teoriassa pienen tarkkuustason tapaus toteuttaa parilla eri tavalla. Yksi vaihtoehto on luoda koko kolmiulotteisesta volumetrisesta datasta epätarkka versio sovellusvälimuistiin heti, ja tästä voidaan reaaliajassa lukea pienen tarkkuustason poikkileikkaus. Toinen vaihtoehto on lukea suoraan pienen tarkkuustason versio poikkileikkauksesta massamuistista. Jälkimmäisessä vaihtoehdossa piilee ongelma liittyen siihen, että kuvasekvenssin yksittäisen mittausarvon $S(x,y,z)$ lukemiseksi, on massamuistista luettava kokonaan yksittäinen kuvasekvenssin kuva. Suurien volumetristen datojen tapauksissa yksittäisetkin kuvat ovat niin suuria, että niiden lataaminen muistiin reaaliaikaisesti ei ole mahdollista. Tällöin pienen tarkkuustason ratkaisu, jossa pidetään epä-

tarkkaa versiota volumetrisesta datasta muistissa, tarjoaa eri tuettavien tiedostomuotojen takia paremman vaihtoehdon pienen tarkkuustason toteuttamiseksi. Pienenä haittapuolena tässä on se, että pienen tarkkuustason malli joudutaan tekemään alussa erikseen ja se vie hieman sovellusvälimuistia.

Pienen tarkkuustason malli voidaan rakentaa samalla logiikalla, mitä käytetään pienen tarkkustasojen poikkileikkausten lukemiseen. Suurentamalla interpolointipisteiden väliä, voidaan luoda pienen tarkkuustason versio poikkileikkauksesta ja volumetrisesta datasta. Jos esimerkiksi interpolointiväliä kaksinkertaistetaan, tällöin kaksiulotteinen poikkileikkaus vie neljännesosan alkuperäisen poikkileikkauksen koosta, kun taas kolmiulotteinen pienen tarkkuustason malli vie kahdeksannesosan alkuperäisen volumetrisen datan koosta. Koska ohjelmamoduuli on suunnattu toimivaksi perustietokoneilla, on hankala määrittää absoluuttista arvoa sille, kuinka paljon sovellusvälimuistia tulisi varata pienen tarkkuustason mallille. Tämän takia varatun muistin määrää on hyvä pystyä vaihtamaan käyttäjän toimesta. Oletetaan, että volumetrisen datan leveys on L, korkeus on K, syvyys on S ja yhden mittausarvon esittämiseen vaadittujen tavujen määrä on T. Tällöin tiedetään, että ilman pakkausalgoritmia volumetrinen datan esittämiseen tarvitaan M määrä tavuja, jossa $M = LKST$. Jotta käyttäjä voi itse määrätä kuinka paljon muistia M_p pienen tarkkuustason versio vie, on datan leveys, korkeus ja syvyys kerrottava interpolointiskaalan kertoimella i. Tällöin voidaan ratkaista interpolointiskaalan kerroin i seuraavasti:

$$M_p = LiKiSiT$$
$$i^3 = \frac{M_p}{LKST}$$
$$i = \sqrt[3]{\frac{M_p}{LKST}}$$

Tällä tavalla saadaan kerroin, jonka avulla voidaan kertoa volumetrisen datan dimensiot, jotta saadaan vastaavat dimensiot pienen tarkkuustason mallille niin, että pienen tarkkuustason malli pysyy käyttäjän määrittelemässä muistinkäyttörajoituksissa. Rajoittamalla i välille $0 < i \leq 1$ voidaan vielä varmistaa, että pienen tarkkuustason mallia ei interpoloida lähtödataa tarkemmaksi. Oletetaan, että volumetrisen datan mittaustulosten väli ja interpolointiväli ovat

1. Tällöin pienemmän tarkkuustason interpolointiväli saadaan i käänteisluvusta. Voidaan siis todeta, että i arvon noustessa interpolointipisteiden välit pienenevät. Tällöin voidaan dynaamisesti tarkentaa pienen tarkkuustason mallista suoraan luettuja poikkileikkauksia interpoloimalla alkuperäisestä tiedostosta poikkileikkausta kertomalla i jollain luvulla. Esimerkiksi kertomalla i kahdella, poikkileikkausta interpoloidaan kaksi kertaa tiheämmin molemmissa ulottuvuuksissa, jolloin poikkileikkauksen muodostamiseen interpoloidaan neljä kertaa enemmän pisteitä. Poikkileikkauksen tarkentamista voidaan jatkaa, kunnes haluttu tarkkuus on saavutettu.

Han (2013) kertovat, että interpoloinnissa käytetyn menetelmän vaikutus kuvanlaatuun tulee esille vasta, kun kuvaa interpoloidaan lähtödataa suuremmalla skaalalla. Tämän takia voidaan tehdä hypoteesi, että kaikkien menetelmien kuvanlaatu pysyy suhteellisen samana interpolointimenetelmästä riippumatta, kunnes interpolointiskaala on yli yhden, eli poikkileikkausta interpoloidaan tiheämmin kuin lähtödatassa on mittaustuloksia.

Ohjelmistomoduulin kehityksen yhteydessä huomattiin myös, että joidenkin datojen tapauksessa, jopa yksittäinen poikkileikkaus saattaa kasvaa niin suureksi, että jos interpolointia toteutetaan täysin samassa mittasuhteessa lähtödatan kanssa, sitä ei pystytä sulavasti renderöimään reaaliajassa kaikilla tietokoneilla. Tällöin olisikin hyvä, että käyttäjä voisi itse määrittää miten tarkaksi ohjelmistomoduuli lopulta tarkentaa dynaamisesti poikkileikkausta, jolloin käyttäjä voisi vähemmän tehokkaammalla tai erittäin suurilla datoilla pienentää interpolointiskaalaa, johon asti ohjelma dynaamisesti tarkentaa poikkileikkauksia. Vaikka tämä tuleekin kuvanlaadun kustannuksella, arvo on silti täysin käyttäjän määriteltävissä. Tällöin käyttäjä voi itse päättää, minkälaisen kompromissin hän tekee suorituskyvyn ja kuvanlaadun välillä. Tällä tavalla käyttäjä voi myös mahdollisesti nostaa interpolointiskaalaa yli yhden, jolloin ohjelmistomoduuli interpoloi poikkileikkauksia lopulta tarkemmaksi kuin mitä alkuperäinen volumetrinen data on. Tällöin käyttäjä voi saada tarkemman poikkileikkauksen, suorituskyvyn kustannuksella, jos hän haluaa. Tässä on huomioitava, että on käytettävä interpolointimenetelmää, joka voi tuottaa lähtödataa pienemmillä interpolointiväleillä uusia arvoja mittaustulosten arvojen välille. Esimerkiksi lähimmän naapurin interpolointimenetelmä ei tuota 1 suuremmalla interpolointiskaalalla tarkempaa kuvaa, koska kaikki sen tuottamat arvot ovat suoraan otettu lähtödatasta.

Alaluvussa 2.2 käytiin läpi miten lähimmän naapurin interpolointi toimii. Kaufman (2000) kertoo, että lähimmän naapurin interpoloinnissa volumetristä dataa käsitellään käytännössä vokseleina, eli jokainen mittaustuloksen pisteen arvo $S(x,y,z)$ määrittää myös kokonaisen ympäröivän alueen. Toteutuksessa tämä tarkoittaa sitä, että on laskettava poikkileikkauksen jokaista pikseliä lähin mittaustulos, jonka arvo määrittää luodun poikkileikkauksen pikselin värin. Koska toteutuksessa volumetristä dataa esittävä alue on muodostettu niin, että volumetrisen datan mittaustulosten arvot muodostavat kolmiulotteisen luonnollisten lukujen määrittämän avaruuden, interpoloitavan pisteen arvo voidaan saada pyöristämällä kaikki kolme pääakselin koordinaattia lähimpään luonnolliseen lukuun. Pyöristettyä koordinaattia vastaava mittaustuloksen arvo on lähimmän naapurin interpolointimenetelmän tuottama arvo $f(x,y,z)$.

Seuraava tutkielmassa toteutettava interpolointimenetelmä on kolmilineaarinen interpolointi. Luvussa 2.2 kerrotaan, että kolmilineaarisessa interpoloinnissa pisteen arvo $f(x,y,z)$ saadaan ottamalla painotettu keskiarvo pistettä ympäröivien kahdeksan pisteen mittaustulosten arvoista $S(x,y,z)$. Mittaustulosten painoarvo riippuu siitä kuinka lähellä ne ovat interpoloitavaa pistettä. Mitä lähempänä mittaustulos on interpoloitavaa pistettä, sitä suurempi painoarvo mittaustuloksen arvolla on. Csébfalvi (2019) kertoo yksiulotteisen lineaarisen interpoloinnin toimivan pisteelle x seuraavasti:

$$f(x) = f(\lfloor x \rfloor)(\lfloor x \rfloor + 1 - x) + f(\lfloor x \rfloor + 1)(x - \lfloor x \rfloor)$$

Csébfalvi (2019) kuvailee, että kolmilineaarinen interpolointi voidaan toteuttaa interpoloimalla lineaarisesti erikseen jokaisen pääakselin suuntaisesti. Tässä tutkielmassa toteutetun ohjelmistomoduulin tapauksessa, pääakselien interpoloinnit ovat toteutettu järjestyksessä X, Y, Z. Olettaen että interpoloitava piste on (x_i, y_i, z_i), tällöin voidaan todeta, että ympäröivien kahdeksan mittaustuloksen arvojen $S(x,y,z)$ pisteiden koordinaatit ovat:

$$x_{i0} = \lfloor x_i \rfloor \quad y_{i0} = \lfloor y_i \rfloor \quad z_{i0} = \lfloor z_i \rfloor$$
$$x_{i1} = \lfloor x_i \rfloor + 1 \quad y_{i1} = \lfloor y_i \rfloor + 1 \quad \text{ja} \quad z_{i1} = \lfloor z_i \rfloor + 1$$

Tällöin saadaan muodostettua interpolointifunktion $f(x_i, y_i, z_i)$ interpoloimalla lineaarisesti jokaisen pääakselin suuntaisesti pistettä (x_i, y_i, z_i) ympäröivien kahdeksan mittaustulosten arvojen välillä, aloittaen X-akselista seuraavasti:

$$X_0 = S(x_{i0}, y_{i0}, z_{i0})(x_{i1} - x_i) + S(x_{i1}, y_{i0}, z_{i0})(x_i - x_{i0})$$
$$X_1 = S(x_{i0}, y_{i1}, z_{i0})(x_{i1} - x_i) + S(x_{i1}, y_{i1}, z_{i0})(x_i - x_{i0})$$
$$X_2 = S(x_{i0}, y_{i0}, z_{i1})(x_{i1} - x_i) + S(x_{i1}, y_{i0}, z_{i1})(x_i - x_{i0})$$
$$X_3 = S(x_{i0}, y_{i1}, z_{i1})(x_{i1} - x_i) + S(x_{i1}, y_{i1}, z_{i1})(x_i - x_{i0})$$

Tästä saadaan neljä väliarvoa, joiden välillä interpoloidaan Y-akselin suuntaisesti seuraavalla tavalla:

$$Y_0 = X_0(y_{i1} - y_i) + X_1(y_i - y_{i0})$$
$$Y_1 = X_2(y_{i1} - y_i) + X_3(y_i - y_{i0})$$

Lopuksi interpolointi toteutetaan Y-akselin suuntaisesta lineaarisesta interpoloinnista saatujen kahden arvon välillä Z-akselin suuntaan, jolloin saadaan ratkaistua kolmilineaarisen interpoloinnin arvo $f(x_i, y_i, z_i)$ seuraavasti:

$$f(x_i, y_i, z_i) = Y_0(z_{i1} - z_i) + Y_1(z_i - z_{i0})$$

Viimeinen tutkielmassa toteutettava interpolointimenetelmä on luvussa 2.2 kuvailtu kolmikuutio interpolointi. Kolmikuutio interpoloinnissa ideana on luoda kolmannen asteen polynomifunktio, joka määrittää pisteen arvon $f(x, y, z)$. Tämä funktio muodostetaan yleensä joka avaruuden ulottuvuuden suuntaan neljästä lähimmästä pisteestä. Kolmiulotteisessa avaruudessa tämä tarkoittaa, että interpoloitavan pisteen arvo lasketaan $4 \times 4 \times 4 = 64$ lähimmän mittaustuloksen $S(x, y, z)$ arvojen avulla. Kolmannen asteen polynomifunktion muodostamiseen on monia eri menetelmiä. Arata (1995) esittää yhdeksi vaihtoehdoksi Catmull-Rom-

menetelmän, joka on yleisesti käytetty menetelmä mutkien approksimointiin. Catmull-Rom-menetelmä on tämän takia valittu käytettäväksi tämän tutkielman toteutuksessa. Arata (1995) kuvailee yksiulotteista Catmull-Rom-menetelmää seuraavasti, jossa $p_{i-1}, p_i, p_{i+1}, p_{i+2}$ ovat peräkkäiset mittaustulosten arvojen pisteet ja $u \in [0,1]$ esittää interpoloitavan pisteen etäisyyttä mittaustuloksen arvon pisteeseen p_{i+1}.

$$C(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{bmatrix}$$

Kuten kolmilineaarisessa interpoloinnissa, niin myös kolmikuutio interpoloinnissa voidaan jokainen ulottuvuus interpoloida erikseen. Merkitään interpoloitavaa pistettä (x_i, y_i, z_i) ja u arvoja $\Delta x = x_{i+1} - x_i$, $\Delta y = y_{i+1} - y_i$ ja $\Delta z = z_{i+1} - z_i$. Tällöin voidaan ensiksi interpoloida 16 arvoa X-akselin suuntaisesti seuraavasti:

$$X_0 = \begin{bmatrix} \Delta x^3 & \Delta x^2 & \Delta x & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} S(x_{i-1}, y_{i-1}, z_{i-1}) \\ S(x_{i0}, y_{i-1}, z_{i-1}) \\ S(x_{i+1}, y_{i-1}, z_{i-1}) \\ S(x_{i+2}, y_{i-1}, z_{i-1}) \end{bmatrix}$$

$$X_1 = \begin{bmatrix} \Delta x^3 & \Delta x^2 & \Delta x & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} S(x_{i-1}, y_{i0}, z_{i-1}) \\ S(x_{i0}, y_{i0}, z_{i-1}) \\ S(x_{i+1}, y_{i0}, z_{i-1}) \\ S(x_{i+2}, y_{i0}, z_{i-1}) \end{bmatrix}$$

$$\vdots$$

$$X_{15} = \begin{bmatrix} \Delta x^3 & \Delta x^2 & \Delta x & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} S(x_{i-1}, y_{i+2}, z_{i+2}) \\ S(x_{i0}, y_{i+2}, z_{i+2}) \\ S(x_{i+1}, y_{i+2}, z_{i+2}) \\ S(x_{i+2}, y_{i+2}, z_{i+2}) \end{bmatrix}$$

Tämän jälkeen saatua kuuttatoista arvoa voidaan interpoloida Y-akselin suuntaisesti seuraavalla tavalla, saaden neljä välitulosta:

$$Y_0 = \begin{bmatrix} \Delta y^3 & \Delta y^2 & \Delta y & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

$$\vdots$$

$$Y_3 = \begin{bmatrix} \Delta y^3 & \Delta y^2 & \Delta y & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_{12} \\ X_{13} \\ X_{14} \\ X_{15} \end{bmatrix}$$

Lopuksi näiden neljän välituloksen avulla saadaan interpoloitu arvo, interpoloimalla välitulokset Z-akselin suuntaisesti seuraavasti:

$$f(x_i, y_i, z_i) = \begin{bmatrix} \Delta z^3 & \Delta z^2 & \Delta z & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}$$

Luvussa 2.2 tuotiin vielä yksi yksityiskohta esille kolmikuutio interpoloinnista. Kuten yllä olevista kaavoista voi nähdä, kolmikuutio interpoloinnissa käytetään jokaisen ulottuvuuden kohdalla kahta mittaustuloksen arvoa interpoloitavan pisteen molemmilta puolilta. Tämä tarkoittaa sitä, että jokaisessa suunnassa interpoloitavan pisteen ympärillä on oltava vähintään kaksi mittaustuloksen arvoa, jotta interpolointimenetelmää voidaan käyttää. Tämä johtaa siihen, että kolmikuutio interpoloinnissa ei voida käyttää volumetrisen datan reunapisteiden tapauksessa. Arata (1995) kertoo, että yleinen käytäntö reunapisteiden tapauksissa on käyttää jotakin muuta interpolointimenetelmää. Tässä tutkielmassa toteutettavan ohjelmistomoduulin tapauksessa volumetrisen datan reunapisteiden interpolointiin käytetään kolmilineaarista interpolointia.

Lekien ja Marsden (2005) kertovat tavasta optimoida kolmikuutio interpoloinnin toteutusta tallentamalla mittaustuloksien pisteiden yhteyteen pisteen funktion kertoimia, sitä mukaan, kun niitä lasketaan. Tässä tutkielmassa toteutettavan ohjelmistomoduulin kohdedata koostuu kuitenkin yleensä niin useasta mittaustuloksen arvosta, että pisteisiin liitettyjen funktion kertoimien pitäminen sovellusvälimuistissa ei ole yleensä mahdollista.

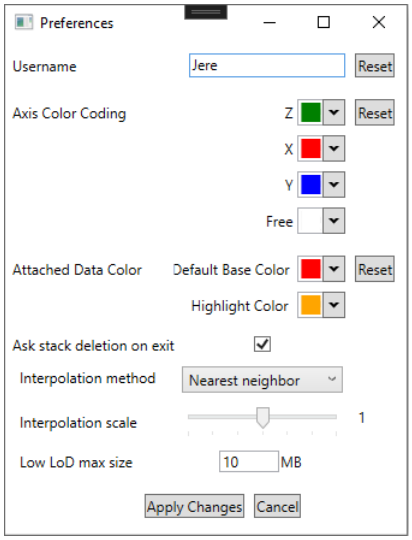
4.4 Integrointi GeoLabiin

Ohjelmistomoduulin valmistuttua toteutuksessa siirrytään integroimaan moduulia luvussa 2.5 kuvailtuun GeoLab-ohjelmistoon. Koska ohjelmistomoduuli ja GeoLab ovat molemmat toteutettu WPF-käyttöliittymäkirjastolla, voidaan moduuli liittää suoraan GeoLabiin erillisenä komponenttina. GeoLabissa käytetään MVVM (Model-View-ViewModel) ohjelmistorakennetta. Anderson (2012) kuvailee MVVM arkkitehtuurin toimivan niin, että näkymä ja taustalla olevat datarakenteet eivät suoraan kommunikoi toistensa kanssa, vaan niiden välissä on näkymämalli, jonka tehtävänä on hoitaa käyttöliittymän kautta saatua käyttäjän syötettä ja ohjata mallin toimintaa syötteen mukaan. Näkymä myös saa näkymämallista käyttöliittymään esitettäviä arvoja, jolloin niitä ei lueta suoraan mallista. Tämä vähentää riskejä tuottaa tahattomasti näkymän puolella esimerkiksi datan esityksen formatoinnissa muutoksia mallin ylläpitämään tietorakenteeseen.

Useat toteutetun ohjelmistomoduulin tarvitsemista parametreista löytyvät jo GeoLabista valmiina. Nämä parametrit ovat:

- Volumetrisen datan leveys
- Volumetrisen datan korkeus
- Volumetrisen datan syvyys
- Volumetrisen datan formaatti
- Tiedostosijainti
- Nimisuodatin
- Tavusyvyys
- Tavujärjestys
- Näytettävä harmaasävyskaala

Sitomalla parametrit näkymämallin riippuvuusominaisuuksiin (eng. dependency property), arvot päivittyvät automaattisesti toteutettuun ohjelmistomoduuliin, niiden muuttuessa. Näiden parametrien lisäksi ohjelmistomoduuli tarvitsee parametrina käytetyn interpolointimenetelmän, interpolointiskaalan, sekä enimmäismuistimäärän, mitä pienen tarkkuustason mallille voidaan varata muistia. Nämä kolme arvoa voidaan lisätä näkymämalliin, jolloin ne päivittyvät muiden parametrien kanssa automaattisesti toteutettuun ohjelmistomoduuliin. Nämä arvot kuvailevat ohjelmistomoduulin toimintaa, jolloin luontevin paikka niiden säätämiseksi on erillinen ikkuna, josta käyttäjä voi hallita GeoLabin toimintaan liittyviä asetuksia. Kuva 10 esittää asetusikkuna, johon on päivitetty tutkielmassa toteutetun ohjelmistomoduulin käyttämät asetukset.



Kuvio 10: GeoLabin asetusikkuna.

Kuten kuvasta 4 nähdään, GeoLabin päänäkymässä voidaan tarkastella volumetrista dataa kolmen pääakselin suuntaisesti. Nämä kolme näkymää vievät suurimman osan GeoLabin päänäkymästä. Kun kaikki kolme näkymää ovat yhtä aikaa päällä, pääikkunaan jää tyhjä tila. Tämä tila on otollinen paikka lisätä toteutettu ohjelmistomoduuli. Tällöin ohjelmistomoduuli on visuaalisesti lähellä muita näkymiä, ja päänikkunaan ei jää tyhjää tilaa, jos kaikki näkymät ovat aktivoituna yhtä aikaa.

Integroinnin yhteydessä tuli myös ilmi, että hahmottamista selkeyttäisi, jos pääakselin suuntaisissa näkymissä näytettäisiin viivan avulla, mistä kohtaa ohjelmistomoduulin esittämä poikkileikkaus leikkaa pääakselin suuntaisten näkymien poikkileikkauksia. Ohjelmistomo-

duuliin lisättiin uusia parametreja, jotka kuvaavat mistä syvyydestä X- ,Y- ja Z-akselien poikkileikkaukset ovat otettu. Näiden avulla voidaan laskea pääakselien näkymille kaksi kaksiulotteisen avaruuden pistettä, jotka kuvaavat ohjelmistomoduulin ja pääakselien poikkileikkausten leikkauskohtaa.

Goldman (1990) kuvailee, että kahden tason T_1 ja T_2 leikkauskohtaa esittävä suora voidaan esittää parametrisessä muodossa $L(t) = P + Vt$. P esittää jotakin suoran L pistettä, V on suuntavektori, jonka suuntaisesti suora kulkee ja t on vapaasti valittava parametri, jota muuttamalla voidaan määrittää kaikki suoran pisteet. V saadaan laskettua tasojen T_1 ja T_2 normaalivektorien $\vec{n_1}$ ja $\vec{n_2}$ ristitulosta $\vec{n_1} \times \vec{n_2}$. V laskemisessa on huomioitava, että jos tasot T_1 ja T_2 ovat samansuuntaiset, ne eivät leikkaa toisiaan ikinä, jolloin normaalivektorien ristitulo on 0.

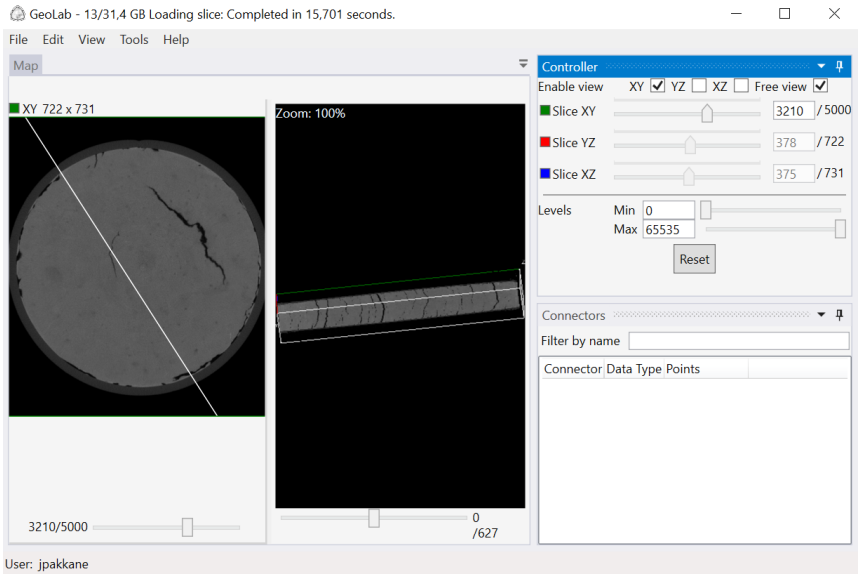
P voidaan laskea määrittämällä kolmas taso T_3, joka ei ole samansuuntainen tasojen T_1 ja T_2 kanssa. Jotta T_3 ei olisi samansuuntainen T_1 ja T_2 kanssa, sen normaalivektori voidaan määrittää tasojen T_1 ja T_2 normaalivektoreiden ristitulona, jolloin $n_3 = \vec{n_1} \times \vec{n_2}$. Tämän lisäksi, T_3 voidaan määrittää niin, että jokin tason pisteistä p_3 sijaitsee origossa, eli pisteessä $(0,0,0)$.

Koska tiedetään, että tasojen T_1, T_2 ja T_3 leikkauspiste on suoralla $L(t)$, tasojen leikkauspistettä voidaan käyttää pisteenä P. Goldman (1990) kertoo, että kolmen tason leikkauspiste voidaan ratkaista tasojen normaalivektorien n_1, n_2 ja n_3 avulla, jos jokaisesta tasosta tiedetään yksi piste, joka sijaitsee kyseisellä tasolla. Koska p_3 on jo määritelty, voidaan tällöin ratkaista P, jos tasoilta T_1 ja T_2 tiedetään niillä sijaitsevat pisteet p_1 ja p_2. Goldman (1990) esittää, että tasojen T_1, T_2 ja T_3 leikkauspiste P voidaan tällöin ratkaista seuraavalla kaavalla:

$$P = \frac{(p_1 \cdot \vec{n_1})(\vec{n_2} \times \vec{n_3}) + (p_2 \cdot \vec{n_2})(\vec{n_3} \times \vec{n_1}) + (p_3 \cdot \vec{n_3})(\vec{n_1} \times \vec{n_2})}{det(\vec{n_1}, \vec{n_2}, \vec{n_3})}$$

Kuvasta 11 nähdään, miltä toteutettu ohjelmistomoduuli näyttää integroituna GeoLab-ohjelmistoon. Kuvassa näkyy, miten Z-akselin suuntaisessa poikkileikkauksessa näytetään sen ja vapaavalintaisen poikkileikkauksen leikkauskohta valkoisena viivana.

Alaluvun 5.3 arvioinnin yhteydessä GTK:lta saadusta palautteesta ilmeni muutama lisäys, joiden toteuttaminen parantaisi ohjelmistomoduulin käytettävyyttä. Nämä lisäykset ovat tar-

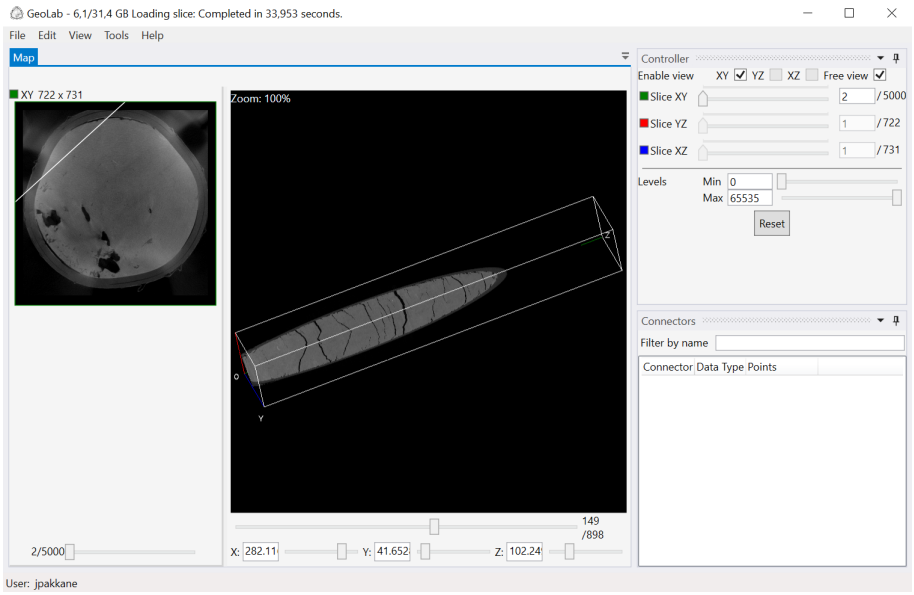


Kuvio 11: Tutkielman ohjelmistomoduuli integroituna GeoLabiin.

kastelukulman säätäminen liukusäätimillä sekä tekstikentillä ja näytettävän poikkileikkauksen tilan tallentaminen GeoLab-projektitiedostoon tallennuksen yhteydessä.

Tarkastelukulman säätäminen liukusäätimien ja tekstikenttien avulla toteutettiin lisäämällä moduulin 3 liukusäädintä sekä niiden viereen tekstikenttä, joiden arvot ovat sidottuina viereisen liukusäätimen arvoon. Jokainen yksittäinen liukusäädin esittää pääakselin suuntaista rotaatiota. Niden kolmen liukusäätimen ja tekstikentän avulla käyttäjä voi säätää tarkastelukulmaa Eulerin kulmien avulla. Jos käyttäjä käyttää kaaripallokameraa, kameran rotaatiot muutetaan vastaaviksi Eulerin kulmiksi, ja arvot päivitetään liukusäätimiin ja tekstikenttiin.

Poikkileikkauksen tilan tallentaminen onnistuu lisäämällä GeoLabin tallennettavaan tietorakenteeseen neljä arvoa. Nämä arvot esittävät Eulerin kulmien X, Y ja Z akselien rotaatiota, sekä poikkileikkauksen syvyyttä. Tarkastelukulman ja syvyyden muutoksien yhteydessä arvot päivitetään samalla GeoLabin datarakenteeseen riippuvuusominaisuuksien avulla. GeoLab-projektin latauksen yhteydessä nämä arvot voidaan myös tuoda samalla tavalla GeoLabin puolelta kehitettyyn ohjelmistomoduuliin. Lopullinen versio tutkielmassa toteutetun moduulin integroinnista GeoLabiin näkyy kuvassa 12. Kuvassa voidaan nähdä vapaan tarkastelukulman näkymän yhteydessä kolme liukusäädintä ja tekstikenttää, joiden avulla käyttäjä voi vaihtaa tarkastelukulmaa.



Kuvio 12: Lopullinen versio ohjelmistomoduulin integroinnista GeoLabiin.

5 Ohjelmistomoduulin arviointi

Tässä luvussa käydään läpi ohjelmistomoduulin eri iteraatioiden arviointia. Arvioinnit ovat jaettu erillisiin alalukuihin, joissa jokaisessa keskitytään luvun 4 alalukujen mukaisten kokonaisuuksien arviointiin. Luvussa 3 käydään läpi yleisellä tasolla tutkimuksessa käytettäviä arviointimenetelmiä. Koska tutkimus tehdään yhteistyössä GTK:n kanssa, heiltä saadaan myös palautetta ohjelmistomoduulin eri iteraatioiden toiminnasta. Tämän palautteen ja arvioinnin pohjalta, artefaktin toimintaa kehitetään tutkielman edetessä.

5.1 Näkymän ohjauksen kehittämisen arviointi

Näkymän ja sen ohjauksen arviointimenetelmänä käytetään heuristista arviointia. Heuristisen arvioinnin tukena käytetään Nielsenin heuristiikkalistausta. Alaluvussa 3.3 käydään tarkemmin läpi miksi näkymää ja sen ohjausta arvioidaan heuristisesti sekä mikä on Nielsenin heuristiikkalista. Nielsenin heuristiikkalistaus sisältää kymmenen eri yleistasoista sääntöä, joita käyttöliittymän tulisi seurata. Seuraavissa kappaleissa käydään ohjelman toimintaa läpi yksittäisten sääntöjen näkökulmista.

Nielsen (1994) kertoo ensimmäinen heuristiikkalistan säännön kertovan, että ohjelman tulisi näyttää käyttäjälleen koko ajan, mikä ohjelman tila on. Ohjelmistomoduuli antaa käyttäjälleen neljä toimintoa, joiden avulla käyttäjä voi ohjata näkymää. Nämä toiminnot ovat kuvakulman vaihtaminen, poikkileikkauksen syvyyden muuttaminen, tarkennustason muuttaminen sekä panorointi. Kuvasta 7 nähdään, että ohjelmistomoduuli näyttää käyttäjälleen avaruuden orientaation oikeasta alakulmasta löytyvällä komponentilla, joka näyttää mistä suunnista pääakselit löytyvät kameran näkökulmasta. Tämä auttaa käyttäjää hahmottamaan tarkastelukulmaa. Poikkileikkauksen syvyys näytetään käyttäjälle numerona liukusäätimen vierestä. Panorointi muuttaa näkymän kohdetta, jolloin sen toiminta näkyy suoraan näkymän muutoksena. Tarkennustasoa ei näy käyttöliittymässä numeraalisesti, joten kyseessä on puute komponentin toiminnassa. Puute on vakavuudeltaan vähäinen, koska tarkennuksen muuttaminen muuttaa myös näkymää, jolloin tarkennustaso on epäsuorasti käyttäjän havaittavissa.

Nielsen (1994) kuvailee toisen heuristisen säännön olevan se, että ohjelma käyttää oikean

maailman termejä. Tarkennettuna tällä tarkoitetaan sitä, että ohjelma ei näytä käyttäjälleen sitä miten ohjelma sisäisesti toimii, vaan käyttäjälle näytetään hänen ymmärrettävissä olevia oikeaan mailmaan rinnastettavia termejä ja näkymiä. Ohjelmistomoduuli näyttää käyttäjälleen suorakulmaisen särmiön, jonka dimensiot vastaavat volumetrisen datan dimensioita. Näiden dimensioiden kanssa samaa asteikkoa käytetään kuvaamaan myös poikkileikkauksen syvyyttä. Tällöin poikkileikkauksen syvyys ja yleinen ohjelman skaala ovat käyttäjälle rinnastettavissa datan dimensioihin. Avaruuden pääakselien suunnat näyttävässä käyttöliittymäkomponentissa lukee pääakselia hahmottavien janojen yhteydessä, mitä akselia janat esittävät. Tämä kertoo tarkastelukulman orientaation käyttäjälle.

Nielsen (1994) kertoo kolmannen heuristiikkalistan säännön kertovan, että käyttäjän kuuluisi olla koko ajan hallinnassa ohjelman toiminnasta. Tämän lisäksi ohjelma ei saisi asettaa käyttäjälle turhia rajoitteita ohjelman toiminnan hallintaan liittyen. Käyttäjä voi aina käyttää kaikkia ohjelmistomoduulin toimintoja, jolloin voidaan todeta, että käyttäjä on täysin hallinnassa ohjelman toimivuudesta. Luvussa 4.2 kerrotaan kuinka kaaripallokameran toteutuksessa X-akselin suuntainen liike on rajoitettu -90° ja 90° välille. Tämä perustuu siihen, että tällöin on helpompi hahmottaa näkymän orientaatiota, koska kameran ylöspäin oleva suunta on aina Y-akselin positiiviseen suuntaan päin. Tämä rajoite ei ole kuitenkaan tarpeellinen, koska ohjelmistomoduulissa on pääakselien suunnat näyttävä komponentti, jonka avulla käyttäjä voi hahmottaa mistä päin kamera tarkastelee volumetristä dataa. Täten kaaripallokameran rajoitteet muodostavat vakavuudeltaan vähäisen käytettävyysongelman.

Nielsen (1994) kuvailee neljännen heuristiikkalistauksen säännön olevan se, että ohjelma käyttää yhtenäisiä termejä sekä ohjelman toiminta on koko ajan yhtenäistä. Ainoat moduulissa käytetyt termit, ovat pääakselien nimet X, Y ja Z sekä käytetty skaala, joka perustuu volumetrisen datan kokoon. Toiminnot ohjelmistomoduulissa ovat yhtenäisiä, ja kuten luvussa 4.2 kerrotaan, ohjelmistomoduulin toimintaa on pyritty myös osin yhtenäistämään GeoLab-ohjelmiston kanssa, johon ohjelmistomoduuli integroidaan.

Nielsen (1994) kertoo heuristiikkalistauksen viidennen kohdan olevan virhetilanteiden esto. Ohjelmistomoduulissa näkymän panorointiin on asetettu rajoite, jonka mukaan kameran kiintopisteen on sijoituttava volumetristä dataa kuvaaman alueen sisälle. Tämä saattoi joissain tilanteissa johtaa siihen, että käyttäjä ei voinut panoroida näkymää halutulla tavalla. Esi-

merkiksi jos poikkileikkauksen syvyyttä on muutettu, kameraa ei välttämättä voinut panoroida niin, että kaikki poikkileikkauksen kohdat olisivat näkyvissä kaikilla tarkennustasoilla. Myös jos panorointia toteutti useasta eri näkökulmasta peräkkäin ilman, että panorointia nollattiin, kamera saattoi jäädä nurkkaan jumiin. Tältä virhetilanteelta voidaan välttyä sillä, että panoroinnin kiintopisteet voivat olla myös volumetristä dataa kuvaavan alueen ulkopuolella. Nykyisessä muodossa panoroinnin rajoitteet muodostavat vakavan käytettävyysongelman.

Nielsen (1994) kuvailee kuudennen heuristiikkalistauksen säännön olevan tunnistaminen muistamisen sijaan. Tämä tarkoittaa, että käyttäjän ei tarvitse ulkoa muistaa mitä jokainen käyttöliittymän komponentit ja eri painike tekevät. Tämä voidaan toteuttaa esimerkiksi niin, että käyttöliittymäkomponenttien muodot viestivät käyttäjälle niiden toiminnoista. Ohjelmistomoduuli on yksinkertainen, ja ainoa toiminnallinen käyttöliittymäkomponentti moduulissa on liukusäädin, jonka avulla poikkileikkauksen syvyyttä voidaan muuttaa. Liukusäätimen yhteydessä lukee myös poikkileikkauksen sen hetkinen syvyys, joka auttaa käyttäjää tunnistamaan mitä liukusäätimellä tehdään. Suurin osa näkymän ohjauksesta tehdään hiirellä, ja parilla yleisellä toimintaa muuttavalla painikkeella. Ainoat suoraan painikkeisiin liitetyt toiminnat ovat Z- ja X-painikkeisiin liitetyt panoroinnin ja poikkileikkauksen syvyyden nollaus. Tällöin ulkoa muistettavien toimintojen määrä pysyy kohtuullisena käyttäjälle.

Nielsen (1994) kertoo heuristiikkalistauksen seitsemännen kohdan olevan joustavuus ja tehokäyttö. Tällä tarkoitetaan sitä, että ohjelmisto on helposti ensikertalaisten käytettävissä, mutta myös mahdollistaa tehokäytön kokeneemmille käyttäjille. Tämä hoidetaan yleensä esimerkiksi pikanäppäinten avulla. Ohjelmistomoduulin toiminta on hyvin yksinkertaista. Shoemake (1994) kertoo että kaaripallokameran pitäisi olla hyvin lähestyttävä ensikertalaiselle. Myös muutkin ohjelmistomoduulin toiminnoista ovat hyvin yksiselitteisiä ja käyttäjäystävällisiä ensikertalaiselle. Tehokäyttöön ohjelmistomoduuli ei varsinaisesti tarjoa työkaluja, mutta tämä myös johtuu siitä, että ohjelmistomoduulin toiminta on yksinkertaista, jolloin varsinaisten tehotoimintojen määrittely itsessään on hankalaa.

Nielsen (1994) kuvailee kahdeksannen heuristiikkalistauksen säännön olevan estetiikka ja minimalismi. Tällä tarkoitetaan sitä, että käyttäjälle ei tarjota ylimääräistä informaatiota, vaan kaikki hänen näkemänsä informaatio on hänelle sillä hetkellä tarpeellista. Ohjelmistomoduuli on toiminnaltaan ja näkymältään hyvin yksinkertainen. Ainoat tiedot mitä moduuli

tarjoaa käyttäjälleen, ovat poikkileikkauksen visualisointi, poikkileikkauksen syvyys ja tarkastelukulma. Tarkastelukulma esitetään käyttäjälle pääakselien suunnat näyttävän komponentin avulla. Poikkileikkauksen visualisointi on koko ohjelmistomoduulin olennaisin näytettävä asia, ja myös poikkileikkauksen syvyys ja tarkastelukulma ovat tärkeitä asioita, jotka kuuluvat olla käyttäjälle aina näkyvissä.

Nielsen (1994) kertoo kahden viimeisen säännön koskevan virhetilanteiden käsittelyä, ja dokumentointia. Nämä eivät kuitenkaan ole tällä hetkellä olennaisia asioita, koska arvioitava kohde on vain ohjelmistomoduuli, eikä kokonainen ohjelmisto. Virheiden hallinta on suoritettava yhtenäisesti sen ohjelman kanssa, johon ohjelmistomoduuli integroidaan, jolloin virheiden hallinta voidaan toteuttaa kunnolla vasta integroinnin yhteydessä. Myös dokumentointi on osa-alue, jossa moduulin toiminnasta kerrotaan osana integroitavan ohjelmiston dokumentointia, jotta yhtenäisyys integroitavan ohjelman ja ohjelmistomoduulin välillä pysyisi. Tällöin kumpaakaan osa-aluetta ei arvioida tämän arvioinnin yhteydessä.

Yhteensä heuristisessa arvioinnissa löydettiin 3 ongelmaa käytettävyyden suhteen, joista 2 olivat vähäisiä ja yksi oli vakava. Nämä ongelmat olivat tarkennustason näyttämisen puute, kaaripallokameran X-akselin suuntaiset rajoitukset sekä panoroinnin rajoittamisen tuomat ongelmat. Heuristisen arvioinnin lisäksi GTK:lta saadussa palautteessa havainnoitiin 4 käytettävyyden ongelmaa. Panoroinnin ja kaaripallokameran rajoitukset olivat samoja ongelmia, jotka löytyivät heuristisessa arvioinnissa. Uutena käytettävyyden ongelmana tuli ilmi, että pääakselien suunnat näyttävä käyttöliittymäkomponentti on ajoittain vaikealukuinen. Toinen uusi löydetty käytettävyyden ongelma oli se, että vaikka panorointi liikuttaakin näkymää oikeaan suuntaan, panoroinnissa hiiri ei pysynyt näkymässä sen kohdan paikalla, josta panorointi aloitettiin. Yhteensä ohjelmistomoduulin arvioinnissa löydettiin 5 ongelmaa käytettävyyteen liittyen. Korjauksien dokumentointi löytyy luvusta 4.2. GTK:n kanssa arvioitiin, että ohjelmistomoduulin ohjauksen ja näkymän käytettävyys on yleisellä tasolla tarpeeksi hyvä, kun nämä viisi käytettävyysongelmaa ovat korjattu. Tällöin toteutuksessa voidaan edetä interpoloinnin toteuttamiseen korjausten jälkeen.

5.2 Volumetrisen datan interpoloinnin arviointi

Interpolointimenetelmien arviointimenetelmänä käytetään dynaamista analyysiä. Luvussa 3.6 käydään tarkemmin läpi miksi interpolointimenetelmiä arvioidaan dynaamisella analyysilla. Interpolointimenetelmien dynaamisen analyysin tarkoituksena on varmistaa, että kaikki kolme toteutettua interpolointimenetelmää tarjoavat käyttäjälle kohtuullisia kompromisseja suorituskyvyn sekä kuvanlaadun välillä.

Olennaisin ero tutkielmassa toteutetun ohjelmistomoduulin ja aiempien olemassa olevien toteutuksien välillä on se, että dataa luetaan dynaamisesti massamuistista sitä mukaan, kun sitä tarvitaan. Tällöin datan lukeminen on oletettavasti pullonkaula suorituskyvyssä. Luvussa 4.3 käydään läpi miten kuvasekvenssin toteutus eroaa raakadatan toteutuksesta. Koska suurin ero kuvasekvenssien ja raakadatan käsittelyn välillä on tapa, jolla dataa luetaan, datasekvenssin ja raakadatan interpolointimenetelmät eivät ole suorituskyvyltään suoraan verrattavissa toisiinsa vertailuanalyysissä. Tämän takia raakadatan ja kuvasekvenssien interpoloinneille toteutetaan erilliset suorituskykyvertailut.

Vokolos ja Weyuker (1998) kertovat, että olennainen osa suorituskyvyn testaamisessa on valita kohdedata joka edustaa hyvin oletettua datajoukkoa. Suorituskykyä testatessa olennaisimmat datan attribuutit, joiden oletetaan vaikuttavat suorituskykyyn, ovat volumetrisen testidatan dimensiot sekä interpoloitavien pisteiden määrä. Tämän takia suorituskykyä on hyvä lähteä testaamaan satunnaisesti generoidulla datalla. Tällöin datan dimensiot voidaan vapaasti määrittää. Visuaalisessa arvioinnissa taas on tärkeää, että volumetrinen data esittää jotakin, eikä ole vain satunnaisesti generoitua. Tällöin on helpompi arvioida eri interpolointimenetelmien välisiä visuaalisia eroja. Tämän takia suorituskykyvertailu ja visuaalinen arviointi toteutetaan erillisillä datoilla.

Vokolos ja Weyuker (1998) kertovat, että olennainen yksityiskohta suorituskykyvertailussa käytettyjen parametrien valinnassa olevan se, että halutaanko arvoja mitata keskivertotilanteesta, vai suorituskyvyltä vaativimman skenaarion tilanteesta. Koska ohjelmistomoduulin käsittelemien volumetristen datojen joukot vaihtelevat suuresti kooltaan, on hankala määritellä keskivertotilannetta. Myös vaativimman mahdollisen skenaarion määrittely on hankalaa, koska varsinaista rajaa volumetrisen datan koolle ei ole ennalta määritelty. GeoLab-

ohjelmisto on suunnattu toimimaan Windows käyttöjärjestelmissä ja suurimpana pullonkaulana toteutuksessa on datan lukeminen dynaamisesti massamuistista. Näiden tietojen avulla voidaan pyrkiä luomaan suorituskyvyllisesti vaativa tapaus volumetrisen datan interpoloinnille, valitsemalla datan dimensiot niin, että yksittäisen arvon lukeminen olisi suorituskyvyllisesti mahdollisimman raskasta.

Karresand, Axelsson ja Dyrkolbotn (2020) kertovat Windows käyttöjärjestelmän käyttävän yleisesti NTFS tiedostojärjestelmää. NTFS tallentaa tiedostot klustereihin, ja yksi klusteri on pienin mahdollinen tila, minkä käyttöjärjestelmä voi varata massamuistista datan tallentamiseen. Yleinen yksittäisen klusterin koko Windows käyttöjärjestelmissä on 4 kilotavua[1]. Yoo ym. (2012) kertovat, että NTFS-menetelmä tallentaa 16 klusteria yhteen joukkoon. Tällöin yhden joukon koko massamuistissa on 64 kilotavua.

Luvussa 2.1 kerrotaan, että tutkielmassa käytetty raakadata on muodossa, jossa data esitetään jonkin pääakselin suuntaisesti. Esim. jos data on esitetty Z-akselin poikkileikkausten suuntaisesti, vierekkäiset X-akselin arvot ovat tiedostossa toistensa vieressä. Tällöin arvot sijaitsevat myös todennäköisesti massamuistissa samassa joukossa sekä samassa klusterissa. Tässä tapauksessa Y-akselin suuntaisesti vierekkäisten arvojen etäisyys toisistaan tiedostossa riippuu datan leveydestä ja yhden arvon tavusyvyydestä. Arvojen etäisyys tällöin on tavusyvyys kerrottuna datan leveydellä. Z-akselin suuntaisesti vierekkäisten arvojen etäisyys riippuu datan leveyden ja tavusyvyyden lisäksi datan korkeudesta. Z-akselin suuntaisesti vierekkäisten arvojen etäisyys toisistaan on tällöin datan leveys kerrottuna datan korkeudella kerrottuna tavusyvyydellä.

Oletetaan, että kahden arvon lukeminen on hitaampaa kahdesta eri klusterijoukosta kuin yhdestä. Tällöin suorituskyvyllisesti vaativa tapaus olisi, että yksi volumetrisen datan rivi veisi yli yhden klusterijoukon verran tilaa massamuistista. Tällöin Y- ja Z-akselien suuntaisesti vierekkäiset arvot joudutaan lukemaan eri klusterijoukoista. Oletetaan testidatan arvojen olevan 16 bittisiä. Tämä tarkoittaisi, että testidatan leveyden pitäisi olla vähintään $(64 \times 1024)B/2B = 32768$ leveä. Tällöin esimerkiksi, jos 16 bittisen testidatan dimensiot ovat $35000 \times 1000 \times 1000$, niin tiedoston kokonaiskoko on 70 gigatavua, joka on myös kokonsa puolesta tarpeeksi suurikokoinen kuvaamaan ohjelmistolla käsiteltäviä tiedostoja.

1. https://learn.microsoft.com/en-us/windows-server/storage/file-server/ntfs-overview

Luvussa 4.3 käytiin läpi miten kuvasekvenssien tapauksessa yhden arvon lukeminen eroaa raakadatan arvon lukemisesta. Yhden arvon lukemiseksi, kokonainen poikkileikkaus, jolla arvo sijaitsee, on luettava massamuistista. Yksittäistä poikkileikkausta voidaan pitää sovellusvälimuistissa niin kauan, kunnes tarvitaan arvo toiselta poikkileikkaukselta. Tämän takia, jos kaikki interpoloitavat pisteet sijaitsevat samalla poikkileikkauksella, niiden interpolointi on jopa nopeampaa kuin raakadatan tapauksessa. Tämä johtuu siitä, että kuvasekvenssin poikkileikkaus voidaan lukea kokonaan yhdellä tiedostonlukuoperaatiolla, ja tämän jälkeen kaikki interpoloimiseen tarvittavat arvot löytyvät valmiiksi sovellusvälimuistista. Koska kuva pidetään välimuistissa, menetelmä vaatii kuitenkin enemmän välimuistia kuin raakadatan tapaus.

Suorituskykyvertailussa ollaan kiinnostuneita vaativasta tapauksesta. Kuvasekvenssin tapauksessa, tämä tarkoittaa sitä, että peräkkäin interpoloitavat pisteet sijaitsevat eri poikkileikkauksissa. Pisteiden interpolointi ohjelmassa suoritetaan poikkileikkauksen näkökulmasta ensin vasemmalta oikealle, ja sen jälkeen ylhäältä alas. Tällöin ottamalla kuvasekvenssin esitysmuodon suuntaisen akselin poikkileikkauksen näkökulmasta vaaka-akseliksi, saadaan testitapaukseksi vaativin tapaus kuvasekvenssille. Koska testitapauksen volumetrinen data on kuvattu Z-akselin suuntaisesti, tarkoittaa tämä sitä, että vaativin tapaus olisi se, että Z-akseli on poikkileikkauksen vaaka-akseli. Tämä saadaan toteutettua esimerkiksi interpoloimalla poikkileikkausta X-akselin suuntaisesti.

Koska kuvasekvenssien lukeminen on huomattavasti hitaampaa kuin raakadatan lukeminen, suorituskykyvertailussa käytetään huomattavasti pienempää volumetrista dataa, jotta testitapaukset saataisiin toteutettua. Käytetty data on kuusitoista bittistä dataa kuten raakadatan tapauksessa, mutta kaikki dimensiot ovat kertaluokkaa pienempiä, eli testidatan koot ovat kuvasekvenssin tapauksessa $3500 \times 100 \times 100$. Käytetty kuvatiedostoformaatti on `TIFF`, joka on yksi yleisimmistä kuvasekvensseissä käytetyistä tiedostomuodoista.

Hypoteesina on se, että tiedostonluku on pullonkaula suorituskyvyssä. Tämä tarkoittaa sitä, että kolmilineaarinen interpolointi on noin 4 kertaa hitaampaa kuin lähimmän naapurin interpolointi. Tämä johtuu siitä, että kolmilineaarisen interpoloinnin tapauksessa käytetään 8 arvoa. Nämä arvot voidaan lukea neljällä tiedostonlukuoperaatiolla, koska vierekkäiset X-akselin suuntaiset arvot ovat tiedostossa vierekkäin. Kolmikuutio interpolointi on samal-

la logiikalla 16 kertaa hitaampi, koska yhden pisteen interpolointiin käytetään 64 läheisen mittaustuloksen arvoa, ja neljä arvoa voidaan lukea yksittäisellä tiedostonlukuoperaatiolla. Kaikki testitapaukset suoritettiin seuraavalla kannettavalla tietokoneella.

- **Malli:** Dell Latitude 5420
- **Suoritin:** 11th Gen Intel Core i5-1135G7
- **Keskusmuisti:** 32 GB
- **Massamuisti:** SN530 NVMe WDC 512GB
- **Käyttöjärjestelmä:** Windows 10 Enterprise

Suorituskykyvertailussa interpoloidaan poikkileikkausta X-akselin suuntaisesti. Raakadatan tapauksessa käytetty volumetrinen data on kooltaan $35000 \times 1000 \times 1000$, ja kuvasekvenssin tapauksessa data on kooltaan $3500 \times 100 \times 100$. Tällöin interpoloitava poikkileikkaus on raakadatan tapauksessa kooltaan 1000×1000, ja kuvasekvenssin tapauksessa poikkileikkaus on kooltaan 100×100. Tällöin raakadatan tapauksessa interpoloidaan 1000000 pistettä, ja kuvasekvenssin tapauksessa 10000 pistettä. Jokainen testitapaus suoritettiin kolmesti, ja taulukossa oleva arvo on kolmen testitapauksen mittaustulosten keskiarvo.

Jokaisen testitapauksen välissä tietokone sammutettiin ja käynnistettiin uudelleen, jotta välimuistissa olevat arvot eivät vaikuttaisi testituloksiin. Testitapauksissa ajanotto alkaa, kun pienen tarkkuustason mallista on luettu epätarkka versio poikkileikkauksesta, ja tarkemman poikkileikkauksen interpolointi massamuistissa olevasta datasta aloitetaan. Ajanotto pysähtyy, kun ohjelma on saanut luettua tarkan version poikkileikkauksesta massamuistista. Ajanotto suoritetaan ohjelmaan lisätyllä ajastimella. Testitapauksissa käytettävä interpolointiskaala on 1, eli poikkileikkauksia interpoloidaan volumetrisen datan natiivilla tarkkuudella. Eri interpolointimenetelmien mittaustulosten keskiarvot ja keskihajonnat löytyvät taulukoista 1, 2 ja 3. Kaikkien testauskertojen mittaustulokset löytyvät liitteestä A.

Raakadatan interpoloinnin testitapausten mittaustuloksissa hypoteesin vastaisesti, kolmikuutio interpolointi on jopa hieman nopeampi kuin kolmilineaarinen interpolointi. Tämä mahdollisesti johtuu käyttöjärjestelmän muistinhallinnasta. Windows käyttöjärjestelmä käyttää muistinhallinnassa tiedostojen kartoitusta (eng. File mapping) (Microsoft 2021). Tiedoston kartoituksessa tiedostojen osia luetaan massamuistista virtuaalimuistiin, jonka kautta käyttö-

Interpolointimenetelmä	Aika (s)	99% keskihajonta
Lähin naapuri	93.424	$\pm 0.288 (\pm 0.31\%)$
Kolmilineaarinen	132.863	$\pm 0.721 (\pm 0.54\%)$
Kolmikuutio	128.767	$\pm 0.023 (\pm 0.06\%)$

Taulukko 1: Raakadatan suorituskykyvertailun mittaustuloksien keskiarvot ja keskihajonnat.

Interpolointimenetelmä	Aika (s)	99% keskihajonta
Lähin naapuri	58.703	$\pm 3.084 (\pm 5.25\%)$
Kolmilineaarinen	109.984	$\pm 2.683 (\pm 2.44\%)$
Kolmikuutio	211.210	$\pm 3.203 (\pm 1.52\%)$

Taulukko 2: Kuvasekvenssin suorituskykyvertailun mittaustuloksien keskiarvot ja keskihajonnat.

järjestelmä pystyy lukemaan kyseisen arvon. Tällöin jo virtuaalimuistista löytyvä arvo voidaan lukea sieltä suoraan, jolloin voidaan välttyä lukemasta arvoa massamuistista. Tällöin virtuaalimuistista löytyvä arvo saadaan luettua huomattavasti nopeammin. Tämän selittäisi miksi esimerkiksi silloin, kun testitapausten välillä tietokonetta ei sammuteta, raakadatan interpoloinnin lähimmän naapurin testitapaus kestää alle viisi sekuntia.

Alaluvussa 4.3 kerrotaan, että ohjelmistomoduuli luo aluksi pienen tarkkuustason mallin, josta voidaan lukea epätarkkoja versioita poikkileikkauksesta reaaliajassa. Kaikissa testitapauksissa pienen tarkkuustason mallin koko oli rajoitettu enimmissään kymmeneen megatavuun. Koska mallin tekemiseen käytetään samaa interpolointimenetelmää kuin mitä poikkileikkausten luomiseen, kolmikuutio interpolointi lukee useamman arvon massamuistista pienen tarkkuustason mallin luomiseen. Tällöin tarkempaa poikkileikkausta luodessa, useampi arvo löytyy jo valmiiksi virtuaalimuistista, jolloin testitapaukseen käytetty aika lyhenee. Tämä selittää miksi kolmilineaarinen interpolointi on suorituskyvyltään lähellä lähimmän naapurin interpolointia, vaikka kolmilineaarinen interpolointi vaatii yhden pisteen laskemiseksi useiden mittaustulosten arvojen lukemista. Kolmilineaarinen interpolointi lukee useamman arvon pienen tarkkuustason mallin luomisen yhteydessä. Tällöin virtuaalimuistista löytyy useamman mittaustuloksen arvo valmiina, jolloin niiden lukemiselta massamuistista välty-

tään.

Koska tulokset erosivat paljon hypoteesista, raakadatan koe suoritettiin uudestaan samalla datalla, mutta eri tarkastelukulmasta. Tällä kertaa poikkileikkaus luettiin Z-akselia kohtisuoraan. Tällöin poikkileikkauksen vaaka-akselina toimii suurikokoinen X-akseli, eli volumetrisen datan leveys, joka on 350000. Uusi testauskerta oli muilta osin identtinen aiemman testauskerran kanssa. Uuden testauskerran tapauksessa interpoloitava poikkileikkaus sisältää $35000 \times 1000 = 35000000$ interpoloitavaa pistettä.

Interpolointimenetelmä	Aika (s)	99% keskihajonta
Lähin naapuri	70.142	$\pm 0.581 (\pm 0.83\%)$
Kolmilineaarinen	288.396	$\pm 4.133 (\pm 1.43\%)$
Kolmikuutio	1261.936	$\pm 21.19 (\pm 1.68\%)$

Taulukko 3: Raakadatan toisen suorituskykyvertailun mittaustuloksien keskiarvot ja keskihajonnat.

Toisella testauskerralla tulokset vastaavat enemmän hypoteesia. Kolmilineaarisessa interpoloinnissa kestää noin 4 kertaa kauemmin kuin lähimmän naapurin interpoloinnissa. Kolmikuutio interpoloinnissa poikkileikkauksen luominen kesti noin 18 kertaa kauemmin. Yksi mahdollinen syy, miksi näin tapahtui, johtuu siitä, että interpoloitavia pisteitä on huomattavasti enemmän. Tällöin massamuistista luettuja arvoja vapautetaan sovellusvälimuistista ajon aikana enemmän, jolloin arvoja joudutaan lukemaan uudestaan massamuistista, eivätkä ne ole suoraan saatavissa välimuistista.

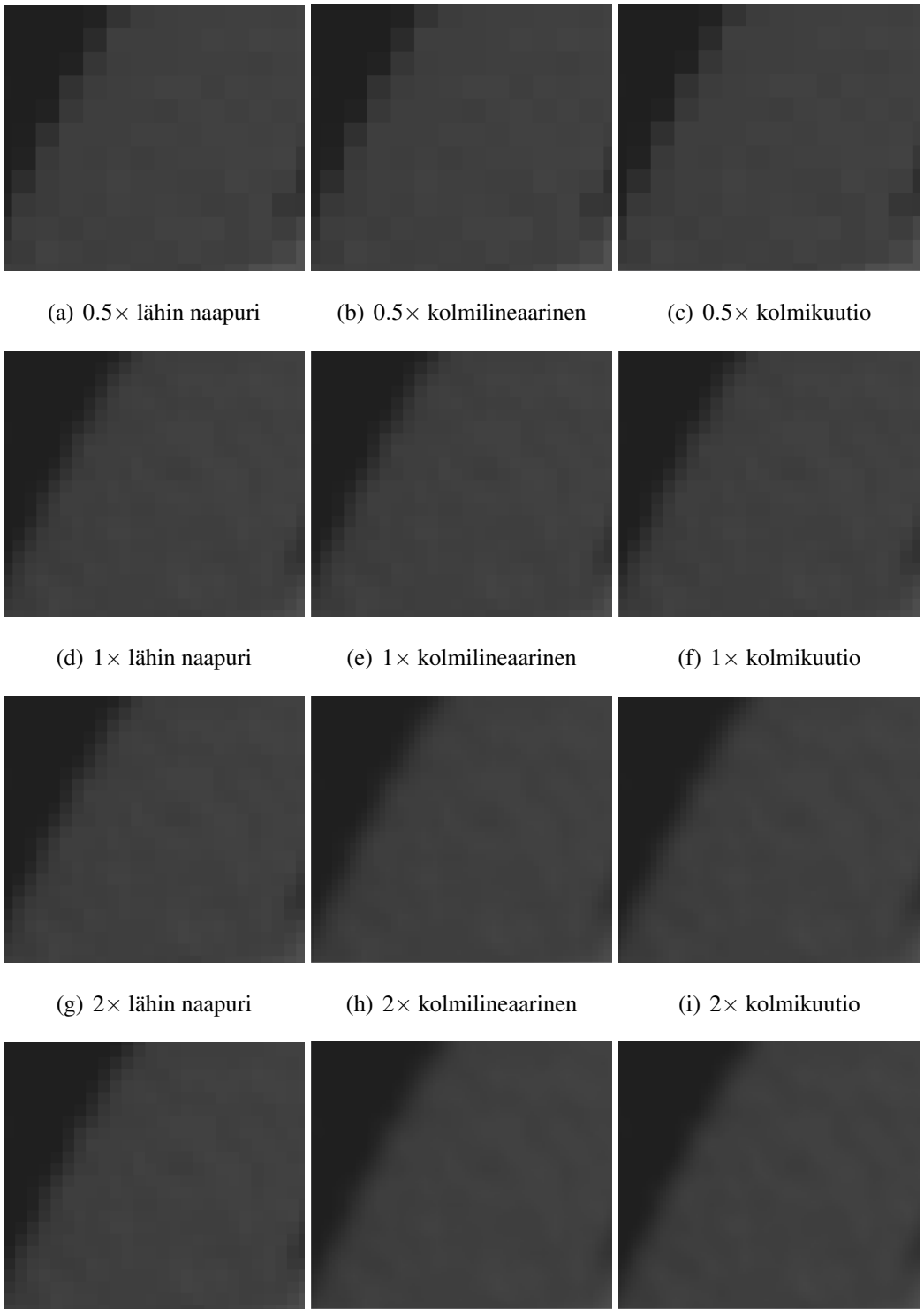
Toinen huomioitava asia toisen testauskerran mittaustuloksissa on se, että lähimmän naapurin interpolointi tuotti poikkileikkauksen huomattavasti nopeammassa ajassa kuin ensimmäisellä testauskerralla, vaikka interpoloitavia pisteitä oli 35 kertaa enemmän. Tämän voidaan olettaa johtuvan siitä, että interpoloitavien pisteiden arvot sijaitsevat tiedostossa vierekkäin, näin ollen vähentäen lukuoperaatioihin kuluvaa aikaa huomattavasti. Tästä voidaan tehdä johtopäätös, että tarkastelukulmalla ja datan dimensioilla on oletettua suurempi vaikutus raakadatan poikkileikkausten luomiseen. Tässä tutkimuksessa ei ole kumminkaan resursseja lähteä tarkemmin selvittämään, minkälainen vaikutus tarkastelukulmalla ja datan dimensioilla on poikkileikkauksen interpoloinnin suorituskykyyn.

Kuvasekvenssien interpolointimenetelmien mittaustuloksissa esiintyi hieman enemmän hajontaa verrattuna raakadatan testauskertojen tuloksiin. Nämä erot johtuvat oletettavasti käytetystä kuvakirjasto. Kuvasekvenssi oli myös hyvin paljon raakadataa hitaampi, kun otetaan huomioon, että kuvasekvenssin käsittelemä data oli 1000 kertaa pienempää kuin mitä raakadatan testitapauksessa ja interpoloitavassa poikkileikkauksessa oli interpoloitavia pisteitä 100 kertaa vähemmän. Vaikka tässä tutkielmassa ei ole resursseja lähteä tarkemmin selvittämään, kuinka kuvasekvenssien lukemista saataisiin optimoitua, tämä on mahdollinen aihe jatkotutkimukselle.

Toinen olennainen osa tutkielmassa suoritettavaa dynaamista analyysiä on visuaalisen tuloksen arviointi. Tällöin on luontevaa, että käytetty volumetrinen testidata esittää visuaalisesti jotakin, eikä ole vain satunnaisesti generoitua dataa. Alaluvussa 2.1 kerrotaan tietokonetomografian olevan yksi yleinen volumetrisen datan tuottamismenetelmä. Koska tutkimuksessa tehdään yhteistyötä GTK:n kanssa, heiltä on saatu tähän tarkoitukseen sopivaa tietokonetomografialla muodostettua testidataa. Testidata, jota käytetään visuaaliseen arviointiin, on osa isompaa dataa, ja on dimensioiltaan $722 \times 731 \times 5000$ kokoinen kuusitoista bittinen volumetrinen data. Raakadatana tiedosto on kooltaan n. 5 gigatavua. Data esittää sylinterimäistä näytettä järvisedimentistä.

Poikkileikkausten visuaalinen arviointi toteutetaan jokaiselle kolmelle eri interpolointimenetelmälle eri interpolointiskaaloilla. Valitut skaalat ovat $0.5\times$, $1\times$, $2\times$ ja $4\times$. Näin voidaan nähdä, miten eri interpolointimenetelmät toimivat natiivilla, pienellä, sekä suurella interpolointiskaalalla. Interpolointimenetelmiä kuvaavat kuvat ovat otettu ohjelmistomoduulista kuvankaappauksilla. Ohjelmistomoduulissa on kuvankaappauksien yhteydessä käytetty $200\times$ tarkennusta. Kuvankaappausten kontrastia on säädetty jälkikäteen kuvien lukemisen helpottamiseksi.

Kuten kuvista 13(a), 13(b), 13(c), 13(d), 13(e) ja 13(f) voidaan nähdä, kaikki menetelmät tuottavat hyvin samankaltaista kuvaa, kun interpolointiskaala $i \leq 1$. Kuvista 13(g), 13(h), 13(i), 13(j), 13(k) ja 13(l) voidaan nähdä, että kolmilineaarinen ja kolmikuutio interpolointi tarjoavat tarkempaa kuvanlaatua silloin, kun $i > 1$. Lähimmän naapurin interpolointi ei taas tarjoa näissä tapauksissa yhtään sen tarkempaa kuvaa kuin silloin jos $i = 1$. Kolmilineaarinen ja kolmikuutio interpolointien visuaaliset erot ovat pieniä, silloin jos $i > 1$.



(a) 0.5× lähin naapuri (b) 0.5× kolmilineaarinen (c) 0.5× kolmikuutio

(d) 1× lähin naapuri (e) 1× kolmilineaarinen (f) 1× kolmikuutio

(g) 2× lähin naapuri (h) 2× kolmilineaarinen (i) 2× kolmikuutio

(j) 4× lähin naapuri (k) 4× kolmilineaarinen (l) 4× kolmikuutio

Kuvio 13: Eri interpolointimenetelmien tuottamat poikkileikkaukset.

Kaikkien interpolointimenetelmien voidaan perustella tarjoavan käytettäviä kompromisseja suorituskyvyn ja kuvanlaadun välillä tilanteesta riippuen. Lähimmän naapurin interpolointi tarjoaa parasta suorituskykyä. Menetelmä tarjoaa myös hyvin samanlaatuista kuvaa muiden menetelmien kanssa silloin, kun $i \leq 1$. Kolmilineaarinen interpolointi taas tarjoaa joissain tapauksissa huomattavasti parempaa suorituskykyä kuin kolmikuutio interpolointi. Tämän lisäksi kolmilineaarinen interpolointi tarjoaa tarkempaa kuvanlaatua kuin lähimmän naapurin interpolointi jos $i > 1$. Lopuksi kolmikuutio interpolointi tarjoaa suhteellisen samankaltaista kuvanlaatua kuin kolmilineaarinen interpolointi. Tämän lisäksi taulukosta 3 nähdään, että kolmikuutio interpolointi suoriutuu poikkileikkauksen tuottamisesta joissain tapauksissa moninkertaisessa ajassa verrattuna kolmilineaariseen interpolointiin.

Toisaalta luvussa 2.2 kerrotaan, että kolmikuutio interpolointi voisi tarjota kolmilineaarista interpolointia parempia visuaalisia tuloksia silloin, jos data ei ole isotrooppista. Tämä voidaan nähdä myös isotrooppisella datalla. Kuvassa 13(k) voidaan nähdä, että kolmilineaarinen interpolointi tuottaa mittaustulosten reunoille rajoja. Kuvasta 13(l) nähdään taas, että näitä rajoja on hankala erottaa kolmikuutio interpoloinnin tuottamasta kuvasta. Tämän takia onkin perusteltavaa, että joissakin tapauksissa käyttäjä voisi haluta käyttää kolmikuutiointerpolointia, sen suuremmista suorituskykyvaatimuksista huolimatta.

Arvioinnin perusteella ohjelmistomoduulin toimintaan voidaan tehdä yksi parannus. Kuvien ja suoritukseen käytettyjen aikojen perusteella voidaan todeta, että lähin naapuri on paras interpolointimenetelmä jos $i \leq 1$. Lähimmän naapurin interpolointi tarjoaa parasta suorituskykyä, ja tilanteissa joissa $i \leq 1$, kuvanlaatu on verrattavissa kolmilineaariseen ja kolmikuutio interpolointiin. Tämän takia lähimmän naapurin interpolointia on hyvä käyttää aina pienen tarkkuustason mallin luomiseen. GTK:n kanssa puhuttiin olisiko tarpeellista käyttää lähimmän naapurin interpolointia myös poikkileikkauksen interpoloinnissa aina jos $i \leq 1$. Tässä päädyttiin kuitenkin siihen tulokseen, että automaattisesti interpolointimenetelmän vaihtaminen mahdollisesti vähentää käyttäjän hallintaa ohjelman toiminnasta, varsinkin jos muutoksesta ei erikseen ilmoiteta käyttäjälle. Tämän takia tätä ei toteutettu tutkielman yhteydessä.

5.3 GeoLabiin integroinnin arviointi

Alaluvussa 3.4 kerrotaan, että moduulin integrointia ohjelmaan arvioidaan kognitiivisella läpikäynnillä. Ensimmäisessä arviointi-iteraatiossa arvioitiin jo ohjelmistomoduulin käytettävyyttä erikseen. Siksi tässä arvioinnissa pyritään arvioimaan, kuinka helposti käyttäjä saa integroidun ohjelmistomoduulin avattua GeoLab-ohjelman kautta. Kognitiivisessa läpikäynnissä läpikäytävä ominaisuus on siis lisätä volumetrinen data GeoLab-ohjelmistoon, ja avata tutkimuksessa toteutettu ohjelmistomoduuli, jossa lisätty data on näkyvissä. Tämän testauksen tarkoituksena on pyrkiä selvittämään kuinka helposti käyttäjä pääsee GeoLabin kautta käyttämään toteutetun ohjelmistomoduulin toimintoja.

Lähtökohtana kognitiiviselle läpikäynnille on kuvassa 14(a) esitetty ohjelman näkymä, joka esitetään käyttäjälle, kun hän avaa ohjelman. Kognitiivisen läpikäynnin tavoite on saada lisättyä volumetrinen data ohjelmaan, ja avata tutkielmassa toteutettu ja integroitu moduuli, jossa on ladattuna GeoLabiin lisätty volumetrinen data. Tavoitteeseen johtavat toiminnot käydään läpi yksitellen, ja jokaisen toiminnon yhteydessä kysytään alaluvussa 3.4 esitetyt neljä kysymystä. Näiden kysymyksien avulla pyritään löytämään mahdollisia käytettävyyden ongelmia, joihin keskiverto käyttäjä saattaisi törmätä.

Kognitiivisessa läpikäynnissä käytetään raakadataa GeoLabiin lisättävän volumetrisen datan formaattina. Raakadata on valittu kognitiivisessa läpikäynnissä käytettäväksi formaatiksi siksi, koska se on hyvin yleinen volumetrisen datan esitysmuoto. Kognitiivisessa läpikäynnissä ohjelman toimintaa pyritään tarkastelemaan siitä näkökulmasta, että käyttäjä ei ole ikinä ennen käyttänyt GeoLabia. Käyttäjältä odotetaan kuitenkin ymmärtävän yleiset volumetrisen datan formaatit, kuten kuvasekvenssi ja raakadata. Koska raakadata ei sisällä datan rakenteen parametreja, vaan käyttäjä joutuu manuaalisesti syöttämään parametrit, raakadatan rakenteen ymmärtäminen on vaatimus, jos käyttäjä työskentelee sen kanssa.

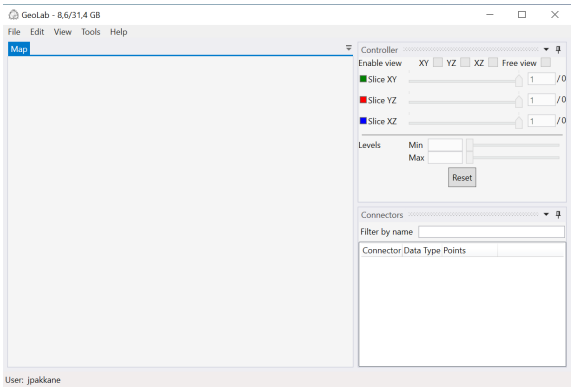
Toiminnot jotka käyttäjän on suoritettava, jotta hän saa ladattua volumetrisen datan GeoLabiin raakadata formaatissa sekä avattua sen näkyviin tutkielmassa toteutetulla ohjelmistomoduulilla ovat seuraavat:

1. Käyttäjä avaa volumetrisen datan lisäysikkunan GeoLabin päänäkymästä.
2. Käyttäjä valitsee volumetrisen datan lisäysikkunasta volumetrisen datan formaatin.

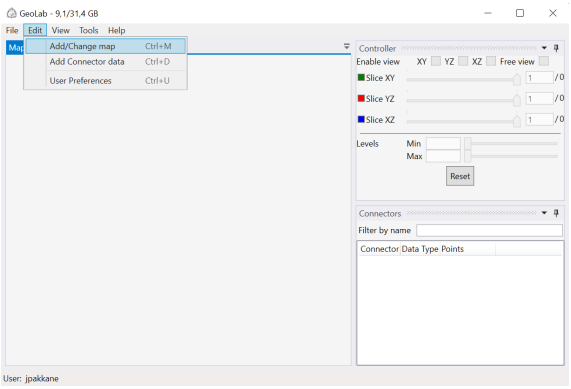
3. Käyttäjä avaa resurssienhallinnan volumetrisen datan lisäysikkunan kautta.
4. Käyttäjä valitsee volumetrisen datatiedoston resurssienhallinnasta.
5. Käyttäjä täyttää volumetrisen datan dimensiot, tavusyvyyden sekä tavujärjestyksen lisäysikkunan kenttiin.
6. Käyttäjä varmistaa volumetrisen datan parametrit.
7. Käyttäjä avaa tutkielmassa toteutetun ohjelmistomoduulin näkyväksi.

Ensimmäinen toiminto, jonka käyttäjän on tehtävä, on lisätä volumetrinen data ohjelmaan. Tämä toteutetaan erillisen volumetrisen datan lisäysikkuna kautta. Lisäysikkunaan pääsee käsiksi kuvassa 14(b) näkyvän `Edit` ylävalikon painikkeen `Add/change map` kautta. Tarkastellaan toimintoa kognitiivisen läpikäynnin kysymyksien avulla.

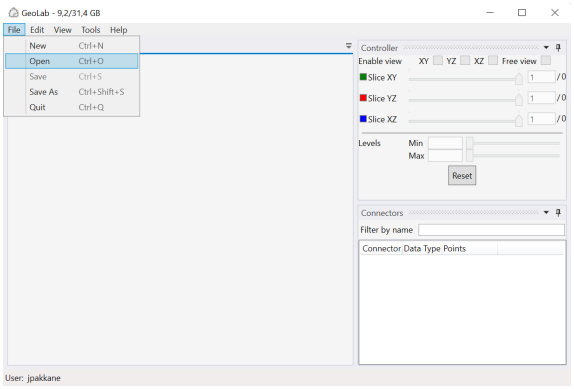
- **Toimiiko käyttäjä oikein saadakseen tavoitteena olevan toiminnon suoritettua?**
- Lisätäkseen volumetrisen datan ohjelmaan, hänen täytyy avata volumetrisen datan lisäysikkuna.
- **Pystyykö käyttäjä löytämään seuraavaa toimintoa?**
- Toiminto löytyy ohjelman ylävalikosta. Vaikka toiminto ei ole heti käyttäjälle esillä, ylävalikon painike löytyy nopeasti, jos käyttäjä lähtee tutkimaan käyttöliittymää omatoimisesti.
- **Pystyykö käyttäjä erottamaan seuraavan toiminnon muista toiminnoista?**
- Iivanainen ym. (2021a) kertovat, että GeoLabissa käytetään termejä `Map` ja `Sample` kuvaamaan eri asioita. `Sample` tarkoittaa GeoLabissa koko projektia, johon kuuluvat volumetrisen datan lisäksi liitedata, liitedatan pisteet sekä projektin metatiedot. `Map` taas tarkoittaa volumetristä dataa, joka toimii karttana, jonka eri kohtiin on liitetty liitedatoja. Tällöin käyttäjä, joka ei tiedä ohjelmassa käytettyä terminologiaa, voi mahdollisesti yrittää lisätä volumetrista dataa kuvassa 14(c) esitetyn `File` ylävalikon painikkeen `Open` kautta. Tämä painike on tarkoitettu aiemmin tallennettujen projektien avaamiseen ohjelmassa. Tällöin on olemassa riski, että käyttäjä yrittää suorittaa väärän komennon volumetrisen datan lisäämiseksi.
- **Saako käyttäjä vastetta jos hän valitsee oikean toiminnon?**
- Kun käyttäjä painaa `Add/change map` -painiketta, hänellä avataan kuvassa 14(d) esitetty uusi ikkuna, jonka avulla hän voi lisätä volumetrisen datan ohjelmaan.



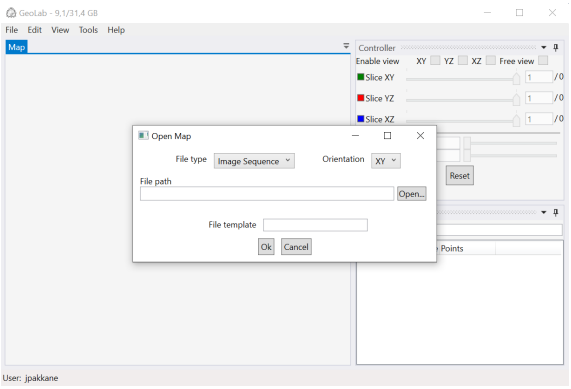
(a) GeoLabin perusnäkymä.



(b) `Add/change map` -painike, jolla aloitetaan volumetrisen datan lisääminen GeoLabiin.



(c) `Open` -painike, jolla avataan GeoLabissa aiemmin tallennettuja projekteja.



(d) GeoLabissa oleva volumetrisen datan lisäämiseen tarkoitettu ikkuna.

Kuvio 14: Volumetrisen datan lisäysikkunan avaaminen GeoLab-ohjelmistossa.

Toisena toimintona käyttäjän on valittava lisättävälle volumetriselle datalle formaatiksi raakadata kuvassa 14(d) esitetyn volumetrisen datan lisäysikkunan alasvetovalikosta, joka on esitetty kuvassa 15(a). Seuraavaksi käydään kognitiivisen läpikäynnin kysymykset dataformaatin vaihdon yhteydessä.

- **Toimiiko käyttäjä oikein saadakseen tavoitteena olevan toiminnon suoritettua?**
- Koska oletuksena on, että käyttäjä ymmärtää dataformaatit, joiden kanssa hän on tekemisissä, hänen seuraava toimintonsa on valita lisättävän datan formaatiksi raakadata.
- **Pystyykö käyttäjä löytämään seuraavaa toimintoa?**
- Volumetrisen datan lisäysikkuna sisältää vain muutaman elementin, joten alasvetova-

likko on helposti käyttäjän löydettävissä.

- **Pystyykö käyttäjä erottamaan seuraavan toiminnon muista toiminnoista?**
- Alasvetovalikko erottuu hyvin muista ikkunan elementeistä, joten käyttäjä voidaan olettaa erottavan se muista ikkunan elementeistä.
- **Saako käyttäjä vastetta jos hän valitsee oikean toiminnon?**
- Kun käyttäjä valitsee alasveto valikosta `Raw` valinnan, lisäysikkuna muuttaa muotoaan kuvassa 15(b) esitettyyn muotoon, jolla käyttäjä voi valita raakadatan parametrit.

Valittuaan raakadatan lisättävän datan formaatiksi, käyttäjän on kolmantena toimintona lisättävä raakadatan sijainti resurssienhallinnan avulla painamalla kuvassa 15(b) esitetyn lisäysikkunan `Open` -painiketta. Tämä avaa kuvassa 15(c) esitetyn resurssienhallinnan, jonka avulla käyttäjä voi valita raakadatan sisältävän tiedoston. Käydään toiminto läpi kognitiivisen läpikäynnin kysymysten avulla.

- **Toimiiko käyttäjä oikein saadakseen tavoitteena olevan toiminnon suoritettua?**
- Jos käyttäjä haluaa tarkastella ohjelmalla volumetrista dataa, hänen on valittava tiedosto, joka sisältää datan.
- **Pystyykö käyttäjä löytämään seuraavaa toimintoa?**
- Volumetrisen datan lisäysikkuna sisältää vain muutaman elementin, joten `Open` painike, jolla resurssienhallinta avataan, on helposti käyttäjän löydettävissä.
- **Pystyykö käyttäjä erottamaan seuraavan toiminnon muista toiminnoista?**
- `Open` -painike on tekstikentän vieressä, jonka päällä lukee `File path:`. Tämän auttaa käyttäjää ymmärtämään, että painikkeen avulla, voidaan valita tiedostosijainti.
- **Saako käyttäjä vastetta jos hän valitsee oikean toiminnon?**
- Kun käyttäjä painaa `Open` -painiketta, kuvassa 15(c) esitetty resurssienhallinta avataan käyttäjälle.

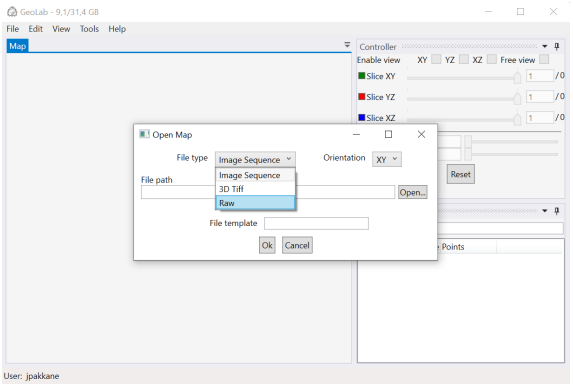
Neljäntenä toimintona käyttäjän on valittava resurssienhallinnasta volumetrisen datan sisältävä tiedosto. Resurssienhallinta on yleisesti käyttöjärjestelmissä käytetty työkalu tiedostojen hallintaan. Sen takia resurssienhallinnan ulkomuoto ja toiminnot vaihtelevat käytetystä käyttöjärjestelmästä riippuen. Koska resurssienhallinta on käyttöjärjestelmän, eikä GeoLabin tarjoama toiminto, tiedoston valitseminen resurssienhallinnasta jätetään käymättä läpi

kognitiivisessa läpikäynnissä.

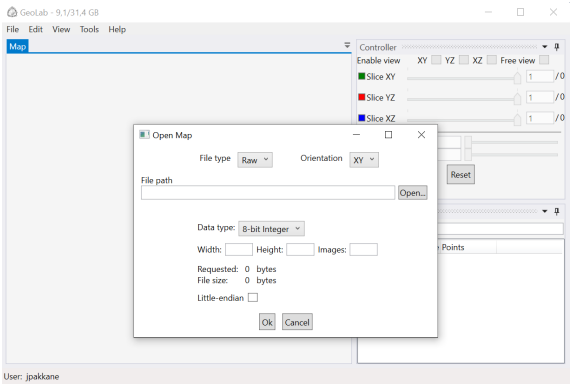
Kun käyttäjä on valinnut lisättävän datan resurssienhallinnasta, viides toiminto, joka käyttäjän täytyy tehdä, on valita raakadatan parametrit. Parametrien valinta hoituu kuvassa 15(d) esitetyn volumetrisen datan lisäysikkunan avulla. Raakadatojen yhteydessä on yleinen käytäntö pitää datan dimensioita mukana tiedoston nimessä. Tämän takia GeoLab yrittää lukea raakadatan nimestä volumetrisen datan dimensiot valmiiksi dimensioiden syöttökenttiin, kun käyttäjä on valinnut haluamansa raakadatatiedoston. Tämä tarkoittaa, että käyttäjän tarvitsee seuraavaksi valita raakadatan käyttämä tavusyvyys sekä tavujärjestys, joten seuraavaksi käydään nämä toiminnot läpi kognitiivisen läpikäynnin kysymysten avulla.

- **Toimiiko käyttäjä oikein saadakseen tavoitteena olevan toiminnon suoritettua?**
- Oletuksena on, että käyttäjä ymmärtää miten raakadata muodostuu. Tällöin hänen voidaan olettaa ymmärtävän, että hänen täytyy valita oikea tavusyvyys sekä tavujärjestys, jotta volumetrinen data näkyy oikein.
- **Pystyykö käyttäjä löytämään seuraavaa toimintoa?**
- Tavusyvyyden valintaan käytettävä alasvetovalikko ja tavujärjestyksen valitsemiseen käytettävä valintaruutu ovat molemmat selvästi käyttäjän nähtävillä.
- **Pystyykö käyttäjä erottamaan seuraavan toiminnon muista toiminnoista?**
- Koska käyttäjä on jo valinnut volumetrisen datan formaatin, käyttäjä luultavasti osaa yhdistää `Data type`-alasvetovalikon raakadatan datasyvyyteen. Varsinkin, kun vakioarvo valikossa on `8-bit integer`, joka on yleinen tavusyvyyden arvo raakadatoille. Tavujärjestyksen valintaruudun vieressä lukee `Little-endian`, joka on toinen mahdollisista tavujärjestyksistä. Tällöin käyttäjän voidaan olettaa ymmärtävän, että jos valintaruutua ei ole valittu, käytetään big-endian tavujärjestystä.
- **Saako käyttäjä vastetta jos hän valitsee oikean toiminnon?**
- Käyttäjän muuttaessa alasvetovalikosta käytettävää tavusyvyyttä, muuttuu vastaavasti tavusyvyydestä ja datan dimensioista riippuvan tekstinkentän `Requested bytes:` arvo. Käyttäjä voi verrata tätä arvoa `File size:` tekstikentän arvoon. Kun nämä arvot ovat samat, käyttäjä voi tästä päätellä, että hän on luultavasti valinnut oikean tavusyvyyden, olettaen että syötetyt raakadatan dimensiot ovat oikeat. Tavujärjestyksen valintaruudun vaihdon yhteydessä käyttäjä voi nähdä kuvassa 15(d) esitetystä valin-

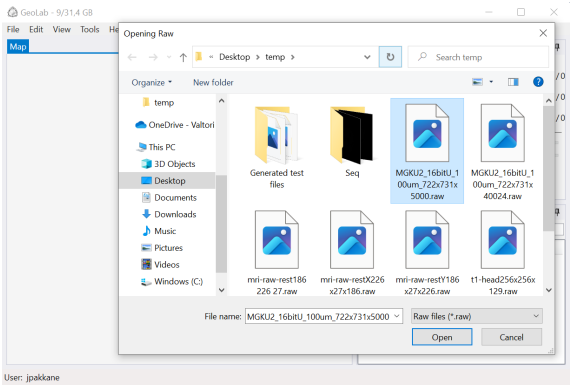
taruudusta, että hän on valinnut little-endian tavujärjestyksen. Käyttäjä ei kuitenkaan tiedä onko arvo oikein, ennen kuin hän pääsee tarkastelemaan tiedostoa valitulla tavujärjestyksellä. Jos käyttäjä käyttää väärää tavujärjestystä, hänen on lisättävä koko volumetrinen data uudestaan ja valita toinen tavujärjestys.



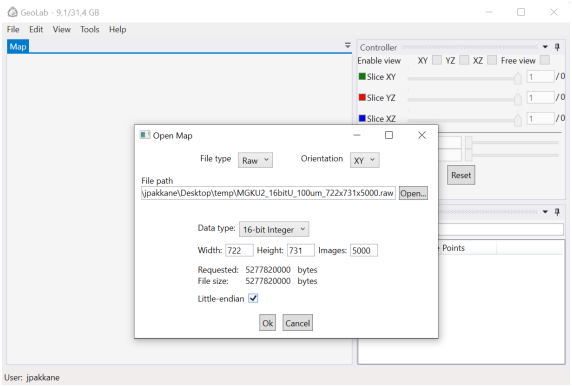
(a) Volumetrisen datan lisäysikkunan formaatin valintaan käytettävä alasvetovalikko.



(b) Volumetrisen datan lisäysikkunan muoto, kun ohjelmassa on valittu raakadata lisättävän datan formaatiksi.



(c) Volumetrisen datatiedoston valitsemiseen käytettävä resurssienhallinta.



(d) Volumetrisen datan lisäysikkuna, johon käyttäjä on syöttänyt raakadatan vaatimat parametrit.

Kuvio 15: Volumetrisen datan ja sen asetuksien valitseminen lisäysikkunassa.

Kun käyttäjä on syöttänyt kaikki raakadatan tarvitsemat parametrit, kuudentena toimintona hän voi lisätä volumetrisen datan ohjelmaan painamalla `Ok`-painiketta. Käydään toiminto läpi kognitiivisen läpikäynnin kysymysten avulla.

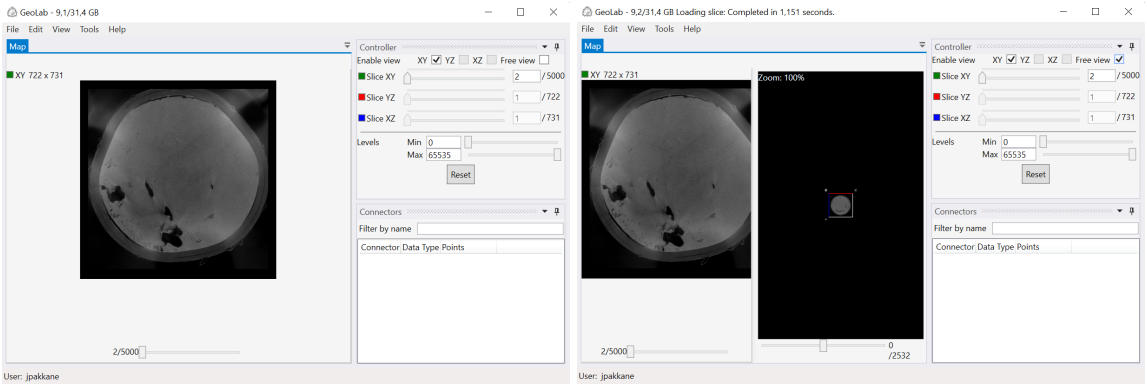
- **Toimiiko käyttäjä oikein saadakseen tavoitteena olevan toiminnon suoritettua?**

- Mitään ei automaattisesti tapahdu, kun käyttäjä on valinnut kaikkien volumetrisen datan kenttien arvot. Tällöin hänen voidaan olettaa tajuavan, että hänen täytyy manuaalisesti varmistaa tiedot oikeaksi painamalla `Ok`-painiketta.
- **Pystyykö käyttäjä löytämään seuraavaa toimintoa?**
- `Ok` -painike on selvästi käyttäjän näkyvissä volumetrisen datan lisäysikkunassa.
- **Pystyykö käyttäjä erottamaan seuraavan toiminnon muista toiminnoista?**
- `Ok` -painike on yleisesti dialogi-ikkunoissa käytetty painike, jolla käyttäjä pääsee eteenpäin. Tällöin toimintoa on hankala sekoittaa mihinkään muuhun ikkunasta löydettävään toimintoon.
- **Saako käyttäjä vastetta jos hän valitsee oikean toiminnon?**
- Kun käyttäjä painaa `Ok` -painiketta, lisäysikkuna sulkeutuu välittömästi, ja käyttäjälle on avattu GeoLabin pääikkunaan kuvassa 16(a) esitetty näkymä, jossa volumetrisen datan esityssuuntainen poikkileikkaus on heti avattuna käyttäjälle.

Seitsemäs ja viimeinen toiminto, jonka käyttäjän täytyy suorittaa, on valita vapaavalintainen näkymä näkyväksi. Tämä tehdään kuvassa 16(a) esitetyn `Free view` valintaruudun avulla. Käydään toiminto läpi kognitiivisen läpikäynnin kysymysten avulla.

- **Toimiiko käyttäjä oikein saadakseen tavoitteena olevan toiminnon suoritettua?**
- Koska käyttäjän tavoitteena on ollut saada vapaan näkökulman näkymä auki, hänen seuraava toimintonsa datan lisäämisen jälkeen on avata vapaan näkökulman näkymä käyttöliittymään.
- **Pystyykö käyttäjä löytämään seuraavaa toimintoa?**
- `Free view` valintaruutu on muiden näkökulmien valintaruutujen kanssa selvästi näkyvillä.
- **Pystyykö käyttäjä erottamaan seuraavan toiminnon muista toiminnoista?**
- `Free view` on ainoa näkökulma, jonka valintaruutu on heti käyttäjän käytettävissä, lisättävän datan suuntaisen näkymän valintaruudun lisäksi. Datan suuntaisen näkymän valintaruutu on jo valmiiksi valittuna, joten tämä viestii käyttäjälle, että `Free view` valintaruudun avulla, hän voi saada näkyviin "vapaan näkymän".
- **Saako käyttäjä vastetta jos hän valitsee oikean toiminnon?**
- Kun käyttäjä on valinnut `Free view` valintaruudun, hänelle avataan heti kuvassa

16(b) esitetysti päänäkymään tutkielmassa toteutettu ohjelmistomoduuli. Moduuliin on ladattuna käyttäjän lisäämä volumetrinen data.



(a) GeoLabin näkymä, kun käyttäjä on lisännyt volumetrisen datan ohjelmaan.

(b) GeoLabin näkymä, kun käyttäjä on valinnut tutkielmassa toteutetun ohjelmistomoduulin näkyväksi.

Kuvio 16: GeoLabin näkymät, kun ohjelmaan on lisätty volumetrista dataa.

Kaikkiaan kognitiivisessa läpikäynnissä löydettiin kaksi käytettävyyden ongelma. Ensimmäinen ongelma on se, että käyttäjän voi sekoittaa ylävalikosta löytävät toiminnot projektin ja volumetrisen datan lisäämiseksi. Tätä ongelmaa voidaan pitää kuitenkin vakavuudeltaan vähäisenä. GeoLabin yhteydessä tuleva käyttöopas avaa myös hyvin ohjelmassa käytettyä terminologiaa, jonka ymmärtäminen auttaa käyttäjää välttämään terminologian aiheuttamaan hämmennystä. Tämän takia tutkimuksen yhteydessä tätä käytettävyyden ongelmaa ei lähdetty korjaamaan.

Toinen löydetty ongelma on se, että jos käyttäjä ei tiedä datan sisältöä etukäteen, hän ei voi tietää, onko hän valinnut oikean tavujärjestyksen raakadatojen tapauksessa, ennen kuin hän pääsee tarkastelemaan datasta otettuja poikkileikkauksia. Tavujärjestyksen vaihtamiseksi, käyttäjän täytyisi ladata data kokonaan uudestaan ohjelmaan. Tämän lisäksi, hänen täytyy myös muistaa, mikä tavujärjestys hänellä oli aikaisemmin käytössä. Toisaalta muut kuvaformaatit hoitavat tavujärjestyksen automaattisesti, jolloin tavujärjestyksen vaihtamisominaisuus saattaisi hämmentää käyttäjiä niiden tapauksessa. Asiasta keskusteltua GTK:n kanssa sovittiin, että koska datan lataamista uudelleen voidaan pitää suhteellisen pienenä käytettävyyden ongelmana, sitä ei lähdetä korjaamaan tämän tutkimuksen puitteissa.

Myös GTK:lta saatiin arviointi-iteraation yhteydessä palautetta ohjelmistomoduulin integroinnista ja sen toiminnasta. He löysivät pari pientä ominaisuutta, joiden toteuttaminen parantaisi ohjelmistomoduulin käytettävyyttä. Antamalla käyttäjälle vaihtoehto säätää tarkastelukulmaa liukusäätimien tai tekstikenttien avulla, käyttäjä pystyisi halutessaan säätämään tarkastelukulmia hyvin tarkasti. Tämän lisäksi vapaan poikkileikkauksen tarkastelukulman sekä syvyyden tallentaminen tarjoaisi mahdollisuuden tallentaa GeoLabin projektitiedostoihin käyttäjän haluaman poikkileikkauksen. Tämä auttaa käyttäjiä jakamaan tallennettujen projektitiedostojen avulla mielenkiintoisia poikkileikkauksia volumetrisesta datasta. Toteutettujen lisäysten dokumentointi löytyy alaluvusta 4.4.

6 Johtopäätökset

Tässä luvussa käydään läpi, mitä menetelmiä tutkimuksessa toteutetussa ohjelmistomoduulissa käytetään, jotta se pystyy täyttämään tutkimuksen alussa sille asetetut kriteerit. Luvussa kiteytetään myös, miten toteutetun ohjelmistomoduulin kehitys eteni sekä miten moduulia arvioitiin. Tämän lisäksi luvussa kerrotaan, mitä mahdollisia jatkotutkimusaiheita tutkimuksen yhteydessä tuli ilmi. Lopuksi luvussa esitetään pohdintaa siitä miten ohjelmassa käytettyjä menetelmiä voitaisiin soveltaa muissa yhteyksissä.

Luvussa 1 kerrotaan, että tutkimuksen tavoitteena on selvittää, miten suurikokoista volumetrista dataa voidaan visualisoida perustietokoneella. Tutkittavaa ongelmaa lähdettiin ratkaiseman suunnittelutieteellisen tutkimuksen kautta. Hevner ym. (2004) määrittelevät suunnittelutieteen olevan iteratiivinen lähestymistapa ongelman ratkaisemiseen, joka pohjautuu kehitys- ja arviointi-iteraatioihin. Tässä tutkielmassa toteutettiin ohjelmistomoduuli, jonka avulla voidaan visualisoida suurikokoista volumetristä dataa perustietokoneella. Tämän lisäksi ohjelmistomoduuli integroitiin alaluvussa 2.5 esitettyyn GeoLab-ohjelmaan. Hevner ym. (2004) kuvailevat tämän tyyppisten tutkimusten lähestyvän ratkaistavaa ongelmaa instantiaation kautta. Instantiaatioiden tarkoituksena on osoittaa ratkaisun toimivuutta ongelman oikeassa esiintymisympäristössä.

Luvussa 2 käytiin läpi eri volumetrisen datan visualisointitekniikoita, joista tutkimuksessa päädyttiin käyttämään viipalointia, jossa volumetristä dataa esitetään poikkileikkausten avulla. Useat volumetristä dataa käsittelevät ohjelmat, kuten Fiji ja GeoLab, käyttävät viipalointia hyväksi visualisoinnissa. Nämä toteutukset kumminkin vaativat joko sen, että volumetrinen data luetaan kokonaan sovellusvälimuistiin, tai poikkileikkauksia luetaan dynaamisesti vain volumetrisen datan esityssuunnassa. Tässä tutkielmassa toteutettiin uusi lähestymistapa viipalointiin, joka yhdistää poikkileikkausten esittämisen vapaasta kuvakulmasta sekä datan lukemisen dynaamisesti. Toteutuksessa hyödynnetään tarkkuustasoja, jotta käyttäjän syötteeseen voidaan antaa visuaalista vastetta reaaliaikaisesti.

Tutkimuksen kaltaisten ratkaisujen toteutuksessa, kannattaa kiinnittää huomiota erityisesti kolmeen olennaiseen asiaan. Miten suurikokoista dataa käsitellään, miten käyttäjä hallitsee

näkymää sekä kuinka näkymän kulma viestitään selvästi käyttäjälle. Jo suunnitteluvaiheessa kannattaa miettiä, mikä on toteutuksen lähestymistapa suurikokoisten datojen esittämiseen. Tutkielmassa hyödynnettiin tarkkuustasoja, jotta ohjelma voi reaaliaikaisesti vastata käyttäjän syötteeseen, ja ajan kanssa myös tarjota tarkan visuaalisen esityksen poikkileikkauksesta.

Toinen tärkeä yksityiskohta, joka toteutuksessa on otettava huomioon, on käyttäjien kontrollien määrittely. Tutkimuksen toteutuksessa käyttäjille annettiin kaksi tapaa muuttaa näkymää. Näistä ensimmäinen on kaaripallokamera, joka on intuitiivinen ja helppokäyttöinen. Toinen tapa muuttaa näkymää on pääakselien suuntaisten rotaatioiden syöttäminen. Rotaatioiden syöttäminen mahdollistaa hyvin tarkan ja hallitun tarkastelukulman määrittelyn, jota pelkkä kaaripallokamera ei tarjoa.

Lopuksi toteutuksessa kannattaa kiinnittää huomiota, miten ohjelma auttaa käyttäjää hahmottamaan sen hetkisen tarkastelukulman. Tutkimuksessa ohjelmistomoduuli esittää kolmiulotteisen näkymän, jossa yhden kulman vastaiset pääakselien suuntaiset data-alueen rajajanat ovat värjätty eri pääakseleille ominaisilla väreillä. Tämä yksinään ei riitä, jotta käyttäjä voisi aina hahmottaa, mistä suunnasta hän tarkastelee poikkileikkausta. Siksi ohjelman integroinnissa käytetään hyväksi kuvissa 11 ja 12 esitettyjä GeoLabin tarjoamia pääakselien suuntaisia poikkileikkauksia. Poikkileikkauksiin piirretään niiden, ja tutkielmassa toteutetun vapaavalintaisen poikkileikkauksen leikkauskohta. Tämän avulla käyttäjän on helppo hahmottaa, mistä suunnasta hänen tarkastelemansa vapaavalintainen poikkileikkaus on otettu.

Luvussa 4.2 kerrotaan, että tutkielmassa kehitettävän ohjelmistomoduulin toteutus aloitettiin käyttöliittymästä. Sen avulla käyttäjä voi vaihtaa tarkastelukulmaa sekä syvyyttä, josta poikkileikkaus esitetään. Sen lisäksi ohjelmistomoduulin käyttöliittymä näyttää käyttäjälleen volumetrista dataa esittävän alueen, jotta käyttäjän on helpompi hahmottaa, mistä kohtaa poikkileikkaus on otettu volumetrisesta datasta. Poikkileikkauksen valinta ja näkymän hallinta toteutettiin tutkimuksessa kaaripallokameran, liukusäätimien sekä näppäin- ja hiirikomentojen avulla.

Luvussa 5.1 käytiin läpi, miten käyttöliittymän käytettävyyttä arvioitiin heuristisella arvioinnilla. Heuristisessa arvioinnissa saatiin selville, että käyttöliittymän käytettävyyttä voidaan parantaa poistamalla kaaripallokameran rajoitteita. Tämän lisäksi selvisi, että yksi yksin-

kertainen ja suhteellisen tehokas tapa auttaa käyttäjää visualisoimaan volumetrisen datan aluetta, on merkitä data-alueen nollakohtaa esittävä kulma jotenkin. Tässä tutkimuksessa se toteutettiin kuvassa 8 esitetyllä tavalla kirjoittamalla 0 nollakohdan kulman viereen. Tästä kulmasta lähtevät sivut värjätään sivun suuntaisen pääakselin ominaisella värillä. Heuristisessa arvioinnissa todettiin myös, että näkymän panorointia ei kannata rajoittaa kauhean tiukasti. Ensimmäisessä käyttöliittymän iteraatiossa panoroinnin kiintopiste oli rajoitettu niin, ettei se voinut poistua volumetrisen data-alueen sisältä. Tämä aiheutti kumminkin ongelmia käytettävyydessä, jonka takia panoroinnin rajoitteita löysättiin arvioinnin jälkeen.

Käyttöliittymän korjausten jälkeen toteutuksessa lähdettiin tekemään poikkileikkausten dynaamista lukemista ja esittämistä. Alaluvussa 4.3 kerrotaan, että alkuperäinen idea datan lukemisen toteutuksessa oli lukea pienen tarkkuustason poikkileikkaus datasta reaaliajassa, ja lähteä tarkentamaan poikkileikkausta taustaprosessissa. Toteutuksen yhteydessä huomattiin kuitenkin ongelma kuvasekvenssien kanssa. Niiden tapauksessa, yhden arvon lukemiseksi, jouduttiin joillakin kuvaformaateilla lukemaan kokonaan yksittäinen kuvasekvenssin poikkileikkaus. Tällöin jopa yhden arvon lukeminen reaaliajassa voisi olla vaikeaa, jos yksittäiset kuvasekvenssin kuvatiedostot ovat suurikokoisia. Tämän takia toteutuksessa päädyttiin tekemään moduulin käynnistyksen yhteydessä pienen tarkkuustason malli koko volumetrisesta datasta. Tästä mallista voidaan lukea reaaliajassa poikkileikkauksia, ja näitä poikkileikkauksia tarkennetaan lukemalla massamuistista taustaprosessissa tarkempaa versiota.

Toinen olennainen osa poikkileikkausten lukemista ja esittämistä oli interpolointi. Toteutuksessa päätettiin toteuttaa kolme yleistä volumetristen datojen visualisoinnissa käytettyä interpolointimenetelmää. Nämä olivat lähimmän naapurin interpolointi, kolmilineaarinen interpolointi ja kolmikuutio interpolointi. Nämä eri menetelmät tarjosivat erilaisia kompromisseja poikkileikkauksen kuvanlaadun ja suorituskyvyn suhteen. Kuvasekvenssien suorituskykyä pyrittiin myös optimoimaan hieman. Tämä toteutettiin pitämällä luettua poikkileikkausta sovellusvälimuistissa, ja lukemalla arvoja siitä, kunnes toisella poikkileikkauksella sijaitsevaa arvoa tarvitaan.

Alaluvussa 5.2 kerrotaan kuinka poikkileikkausten dynaamista lukemista ja esittämistä arvioitiin suorituskykytestauksella ja poikkileikkausten visuaalisella arvioinnilla. Arvioinnin tarkoitus oli varmistaa, että kaikki kolme interpolointimenetelmää tarjoavat käytettäviä kom-

promisseja kuvanlaadun ja suorituskyvyn välillä. Kaikkien kolmen eri interpolointimenetelmien suorituskykyä mitattiin ottamalla aikaa, kuinka nopeasti menetelmät tuottivat poikkileikkauksia. Koska hypoteesina oli se, että raakadata ja kuvasekvenssit eivät ole suorituskyvyllisesti vertailukelpoisia keskenään, molempien suorituskyky testattiin erikseen. Tämän lisäksi eri interpolointimenetelmien tuottamia poikkileikkauksia arvioitiin visuaalisesti.

Suorituskykytestauksen hypoteesi oli, että raakadatan tapauksessa menetelmien käyttämä aika on täysin riippuvainen vain käytetystä tietokoneesta sekä interpoloitavien pisteiden määrästä. Toinen hypoteesi oli se, että kolmilineaarinen interpolointi vie raakadatan tapauksessa noin 4 kertaa kauemmin kuin lähimmän naapurin interpolointi. Hypoteesi perustui siihen, että massamuistista lukuoperaatioita jouduttiin tekemään 4 kertaa enemmän. Samaa logiikka käyttäen, kolmikuutio interpolointi arvioitiin vievän noin 16 kertaa enemmän aikaa, kuin lähimmän naapurin interpolointi, johtuen 16-kertaisesta lukuoperaatioiden määrästä. Kuvasekvenssin suorituskyvystä arvioitiin etukäteen, että tarkastelukulmalla on suuri vaikutus sen toimintakykyyn. Ainoa hypoteesi kuvasekvenssien interpoloinnin toiminnasta oli, että se vie huomattavasti enemmän aikaa, kun raakadatan vastaava tapaus. Tähän poikkeuksena ovat tapaukset, joissa suurin osa interpoloitavista pisteistä sijaitsee samassa kuvatiedostossa. Tämän takia yksi mahdollinen jatkotutkimusaihe olisikin selvittää, miten kuvasekvenssien poikkileikkauksien lukemista mielivaltaisista tarkastelukulmista voitaisiin optimoida.

Taulukoista 1 ja 3 voidaan nähdä, että samalla laitteistolla ja datalla toteutettu testi, jossa käytetään eri tarkastelukulmaa, tuottaa erilaisia tuloksia. Taulukon 3 esittämässä tapauksessa interpoloitavia pisteitä on 35 kertaa enemmän kuin taulukon 1 kuvaamassa tapauksessa. Silti, taulukon 3 tapauksessa, lähimmän naapurin interpolointi tuottaa poikkileikkauksen jopa nopeammin kuin taulukon 1 kuvaamassa tapauksessa. Vaikka kolmilineaarinen ja kolmikuutio interpolointi eivät suoriudu taulukossa 3 esitetystä tapauksesta nopeammin kuin taulukossa 1 esitetystä tapauksesta, ne silti suoriutuvat huomattavasti nopeammin mitä odotettu.

Hypoteesin mukaisesti 35 kertaa suuremmalla interpolointipisteiden määrällä, myös suoritukseen käytetyt ajat olisivat olleet noin 35 kertaa pidemmät. Suorituskyvyn osalta raakadatan tapauksessa taulukossa 1 esitetyt ensimmäiset suorituskyvyn mittaustulokset olivat toisen hypoteesin vastaisia, eli kolmilineaarinen interpolointi vei vain noin 50% enemmän aikaa kuin lähimmän naapurin interpolointi, ja kolmikuutio interpolointi oli jopa hieman no-

peampi kuin kolmilineaarinen interpolointi. Tämän takia koe suoritettiin uudestaan eri tarkastelukulmasta. Tästä saatiin taulukon 3 mukaiset tulokset, jotka alkoivat olla hyvin lähellä hypoteesia. Näiden tulosten perusteella voidaan tehdä johtopäätös, että myös kuvakulmalla on suuri vaikutus poikkileikkausten luomiseen käytettyyn aikaan. Eri kuvakulmien ja datan dimensioiden vaikutuksen selvittäminen poikkileikkausten luomiseen käytettävään aikaan, on mahdollinen jatkotutkimuskysymys. Tutkimuksen tuloksia voidaan mahdollisesti käyttää tässä tutkimuksessa kehitetyn ohjelmistomoduulin optimointiin.

Visuaalinen arviointi perustui kuvaan 13, joka sisältää kuvankaappauksia eri menetelmien tuottamista poikkileikkauksista, eri tarkkuustasoilla. Kuvista 13(g) ja 13(j) voidaan nähdä, että lähimmän naapurin interpolointi, ei tuota tarkempaa kuvaa, jos poikkileikkausta interpoloidaan tiheämmin kuin mitä lähtödatassa on mittaustuloksen arvoja. Toisaalta kuvista 13(b), 13(c), 13(e) ja 13(f) nähdään, että kolmilineaarinen ja kolmikuutio interpolointi eivät tuota visuaalisesti lähimmän naapurin interpolointia tarkempia poikkileikkauksia, jos interpolointiskaala on ≤ 1. Suorituskyvyn ja visuaalisen arvioinnin tuloksena oli se, että kaikki kolme interpolointimenetelmää tarjoavat käytettäviä kompromisseja kuvanlaadun ja suorituskyvyn välillä.

Tutkimuksen kolmannessa kehitysiteraatiossa integroitiin toteutettu ohjelmistomoduuli GeoLab-ohjelmistoon. Alaluvussa 4.4 kerrotaan, että GeoLab sekä ohjelmistomoduuli ovat kehitetty WPF-käyttöliittymäkirjastolla. Tämän takia integrointi oli mutkatonta. Suurin osa ohjelmistomoduulin tarvitsemista parametreista saatiin suoraan GeoLabin tietorakenteesta. Muutaman ohjelmistomoduulin käyttämän asetuksen asettamista varten lisättiin kuvassa 10 näkyvään GeoLabin asetusikkunaan kentät, joiden avulla käyttäjä voi vaihtaa ohjelmistomoduulin parametreja halutessaan. Itse ohjelmistomoduuli integroitiin kuvassa 12 esitetyllä tavalla suoraan GeoLabin pääikkunaan. Tästä ikkunasta käyttäjä voi näyttää tai piilottaa toteutetun moduulin, jos ohjelmaan on ladattu volumetrista dataa.

Integrointia arvioitiin alaluvussa 5.3 kognitiivisella läpikäynnillä. Arvioinnin tarkoitus oli selvittää, kuinka helposti käyttäjä pääsee ohjelman kautta ohjelmistomoduulin tarjoamiin toimintoihin käsiksi. Arvioinnissa löydettiin pari pientä käytettävyyden ongelmaa, mutta niiden katsottiin olevan suhteellisen vähäisiä, jolloin niitä ei lähdetty korjaamaan tämän tutkimuksen yhteydessä. Arvioinnin yhteydessä tuli kumminkin ilmi, että tarkastelukulman tarkka

määrittäminen olisi helpompaa, jos käyttäjä voisi määrittää tarkastelukulman numeerisesti. Tämän takia toteutettuun ohjelmistomoduulin lisättiin kolme tekstikenttää ja liukusäädintä, joiden avulla käyttäjä voi määrittää tarkastelukulman pääakselien suuntaisten rotaatioiden avulla.

Tutkielmassa kehitetty menetelmä suunniteltiin käsittelemään vain yksikanavaista volumetrista dataa. Saman menetelmän pitäisi kuitenkin myös soveltua toimimaan monikanavaisella volumetrisellä datalla. Tutkielmassa kehitetty menetelmä voidaan myös yleistää suurikokoisille visualisoitaville datoille. Suurikokoisesta datasta tehdään ensiksi pienen tarkkuustason versio. Pienestä versiosta voidaan käyttäjän antamilla parametreilla antaa reaaliaikaisesti visuaalinen esitys. Tätä esitystä voidaan lähteä tarkentamaan taustaprosessissa, joka jatkuu, kunnes käyttäjä muuttaa visualisoinnin parametreja, tai ohjelma saa valmiiksi visualisoinnin lopputuloksen. Jos käyttäjä muuttaa visualisoinnin parametreja, otetaan uusi esitys pienen tarkkuustason mallista, ja aloitetaan uuden tarkemman version tuottaminen taustaprosessina. On huomioitava, että yllä kuvailtu menetelmä perustuu siihen, että visuaalisen esityksen luomiseen käytettävä aika on riippuvainen datan koosta. Tämän lisäksi käytettävän visualisointimenetelmän pitäisi pystyä luomaan pienestä datajoukosta visuaalinen esitys reaaliajassa.

7 Yhteenveto

Tässä tutkimuksessa toteutettiin ohjelmistomoduuli, jonka avulla voidaan visualisoida suurikokoista volumetrista dataa. Visualisointi toteutettiin viipaloinnilla, jossa käyttäjä voi valita ja muuttaa poikkileikkauksen tarkastelukulmaa sekä syvyyttä mielivaltaisesti. Ohjelmistomoduuli toteutettiin suunnittelutieteelliselle tutkimukselle ominaisesti useassa eri arviointi- ja kehitysiteraatiossa. Tutkimus toteutettiin kolmessa syklissä, joista kaksi ensimmäistä keskittyivät ohjelmistomoduulin toiminnan kehittämiseen. Viimeisessä kehitysiteraatiossa ratkaisun toimivuus osoitettiin ongelman oikeassa esiintymisympäristössä. Tämä toteutettiin integroimalla moduuli jo olemassa olevaan volumetristä dataa käsittelevään ohjelmaan.

Käytettävyyden osalta tutkimuksessa toteutettiin muutama eri menetelmä, joidenka avulla käyttäjä pystyy säätämään poikkileikkauksen tarkastelukulmaa ja syvyyttä. Luvussa 4.2 kerrotaan, että käyttäjä voi muuttaa tarkastelukulmaa kaaripallokameran raahaamisominaisuuden avulla. Myöhemmin alaluvussa 4.4 kerrotaan, että moduuliin lisättiin myös liukusäätimet sekä tekstikentät, joiden avulla käyttäjä voi myös muuttaa tarkastelukulmaa. Yhdistämällä kaaripallokameran intuitiivisuutta sekä liukusäätimien ja tekstikenttien täsmällisyyttä käyttäjälle annetaan tehokkaat työkalut ohjelmistomoduulin tarkastelukulman hallintaan. Tutkielmassa toteutettiin myös poikkileikkauksen panorointi sekä näkymän tarkentaminen, jotka ovat tärkeät työkalut käyttäjän näkymän hallitsemiseen.

Jotta suurikokoisesta volumetrisesta datasta voitaisiin lukea mielivaltaisia poikkileikkauksia reaaliajassa, päädyttiin tutkimuksessa käyttämään tarkkuustasoja. Tarkkuustasojen avulla voidaan käyttäjälle tarjota reaaliaikaisesti epätarkka poikkileikkaus, jota tarkennetaan taustaprosessina ajan myötä. Alaluvussa 4.3 käsitellään, kuinka tutkimuksen aikana huomattiin, että kuvasekvenssien tapauksessa yksittäisten arvojen lukeminen massamuistista voi viedä niin kauan, että niistä ei voida reaaliaikaisesti muodostaa edes pienen tarkkuustason poikkileikkausta.

Ratkaisuna kuvasekvenssien tapaukseen kehitettiin menetelmä, jonka avulla luodaan pienen tarkkuustason malli koko volumetrisesta datasta, ei vain poikkileikkauksesta, datan lisäämisen yhteydessä, ja pidetään pienen tarkkuustason mallia koko ajan sovellusvälimuistissa.

Mallin luominen vie hetken aikaa, kun ohjelmistomoduuliin lisätään dataa, mutta mahdollistaa myös kuvasekvensseistä pienen tarkkuustason poikkileikkauksen lukemisen reaaliajassa. Poikkileikkauksen muodostamiseksi, tutkimuksessa toteutettiin myös kolme yleisintä volumetrisen datan yhteydessä käytettävää interpolointimenetelmää.

Ohjelmistomoduulin toimintaa arvioitiin kolmeen otteeseen eri arviointi-iteraatioissa. Ensimmäinen iteraatio arvioi käyttöliittymän toimintaa heuristisella arvioinnilla. Toinen iteraatio arvioi eri interpolointimenetelmien tarjoamia kuvanlaadun ja suorituskyvyn kompromisseja ohjelmistomoduulin tilanteessa, jossa dataa luetaan dynaamisesti massamuistista, dynaamisella analyysilla. Viimeinen iteraatio arvioi, kuinka hyvin ohjelmistomoduuli on integroitu ohjelmaan, kognitiivisen läpikäynnin avulla.

Tutkimuksen yhteydessä ilmeni kaksi mahdollista jatkotutkimuskysymystä. Ensimmäinen jatkotutkimusaihe on selvittää, miten kuvasekvenssien poikkileikkausten luomista voidaan optimoida nykyisestä toteutuksesta. Toinen mahdollinen jatkotutkimusaihe on selvittää, miten poikkileikkauksen eri tarkastelukulmat ja syvyydet vaikuttavat poikkileikkauksen luomisen suorituskykyyn. Tätä tietoa voidaan mahdollisesti käyttää optimoimaan nykyistä toteutusta.

Lähteet

Anderson, Chris. 2012. “The model-view-viewmodel (mvvm) design pattern”. Teoksessa *Pro Business Applications with Silverlight 5,* 461–499. Springer.

Arata, Louis K. 1995. “Tricubic interpolation”. Teoksessa *Graphics Gems V,* toimittanut Alan W. Paeth, 107–110. Academic Press. https://doi.org/10.1016/B978-0-12-543457-7.50024-3.

Ball, Thoms. 1999. “The concept of dynamic analysis”. *ACM SIGSOFT Software Engineering Notes* 24 (6): 216–234.

Beutel, Alex, Thomas Mølhave, Pankaj K Agarwal, Arnold P Boedihardjo ja James A Shine. 2011. “TerraNNI: natural neighbor interpolation on a 3D grid using a GPU”. Teoksessa *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* 64–74.

Bourke, Paul. 1999. “Interpolation methods”. Viitattu 1. marraskuuta 2023. https://paulbourke.net/miscellaneous/interpolation/index.html.

Csébfalvi, Balázs. 2019. “Beyond trilinear interpolation: Higher quality for free”. *ACM Trans. Graph.* 38 (4). https://doi.org/10.1145/3306346.3323032.

Drebin, Robert A., Loren Carpenter ja Pat Hanrahan. 1988. “Volume rendering”. *SIGGRAPH Comput. Graph.* (New York, NY, USA) 22 (4): 65–74. https://doi.org/10.1145/378456.378484.

Fadnavis, Shreyas. 2014. “Image interpolation techniques in digital image processing: an overview”. *International Journal of Engineering Research and Applications* 4 (10): 70–73.

Goldman, Ronald. 1990. “Intersection of two lines in three space”. Teoksessa *Graphics Gems,* toimittanut Andrew S. Glassner, 1:304–305. Academic Press.

Goode, Adam, Benjamin Gilbert, Jan Harkes, Drazen Jukic ja Mahadev Satyanarayanan. 2013. “OpenSlide: A vendor-neutral software foundation for digital pathology”. *Journal of Pathology Informatics* 4 (1): 27.

Han, Dianyuan. 2013. “Comparison of commonly used image interpolation methods”. Teoksessa *Conference of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013),* 1556–1559. Atlantis Press.

Hevner, Alan R. 2007. “A three cycle view of design science research”. *Scandinavian Journal of Information Systems* 19 (2): 4.

Hevner, Alan R., Salvatore T. March, Jinsoo Park ja Sudha Ram. 2004. “Design Science in Information Systems Research”. *MIS Quarterly* 28 (1): 75–105.

Hsieh, Jiang. 2003. *Computed Tomography: Principles, Design, Artifacts, and Recent Advances.* SPIE press.

Iinuma, T. 1983. “Principles”. Teoksessa *Illustrated Computer Tomography: A Practical Guide to CT Interpretations,* toimittanut Shinji Takahashi, Sadayuki Sakuma ja Masao Kaneko, 9–22. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-81814-1_3.

Iivanainen, Iiro, Linna Harri, Pakkanen Jere ja Vilavaara Riikka. 2021a. “Groundhog-sovellusprojekti: Sovellusraportti”. Viitattu 1. marraskuuta 2023. https://sovellusprojektit.it.jyu.fi/groundhog/dokumentit/groundhog_sovellusraportti_1.0.0_27102021.pdf.

———. 2021b. “Groundhog-sovellusprojekti: Vaatimusmäärittely”. Viitattu 1. marraskuuta 2023. https://sovellusprojektit.it.jyu.fi/groundhog/dokumentit/groundhog_vaatimusmaarittely_1.0.0_24082021.pdf.

Iivari, Juhani. 2007. “A paradigmatic analysis of information systems as a design science”. *Scandinavian Journal of Information Systems* 19 (2): 39–64.

Jeffries, Robin, James R Miller, Cathleen Wharton ja Kathy Uyeda. 1991. “User interface evaluation in the real world: a comparison of four techniques”. Teoksessa *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 119–124.

Karresand, Martin, Stefan Axelsson ja Geir Olav Dyrkolbotn. 2020. “Disk cluster allocation behavior in windows and ntfs”. *Mobile Networks and Applications* 25:248–258.

Kaufman, Arie E. 2000. “Volume visualization in medicine”. Teoksessa *Handbook of Medical Imaging,* toimittanut Isaac N. Bankman, 713–730. Biomedical Engineering. San Diego: Academic Press. https://doi.org/10.1016/B978-012077790-7/50050-3.

Lee, Seungyong, George Wolberg ja Sung Yong Shin. 1997. “Scattered data interpolation with multilevel B-splines”. *IEEE Transactions on Visualization and Computer Graphics* 3 (3): 228–244.

Lekien, Francois ja J Marsden. 2005. “Tricubic interpolation in three dimensions”. *International Journal for Numerical Methods in Engineering* 63 (3): 455–471.

Levin, Barnaby D.A., Yi Jiang, Elliot Padgett, Shawn Waldon, Cory Quammen, Chris Harris, Utkarsh Ayachit ym. 2018. “Tutorial on the visualization of volumetric data using tomviz”. *Microscopy Today* 26 (1): 12–17. https://doi.org/10.1017/S1551929517001213.

Luebke, David, Martin Reddy, Jonathan D Cohen, Amitabh Varshney, Benjamin Watson ja Robert Huebner. 2003. *Level of Detail for 3D Graphics.* Morgan Kaufmann.

Machiraju, R. ja R. Yagel. 1995. “Accuracy control of reconstruction errors in volume slicing”. Teoksessa *Proceedings 1995 Biomedical Visualization,* 50–57. https://doi.org/10.1109/BIOVIS.1995.528705.

Mahatody, Thomas, Mouldi Sagar ja Christophe Kolski. 2010. “State of the art on the cognitive walkthrough method, its variants and evolutions”. *Intl. Journal of Human–Computer Interaction* 26 (8): 741–785.

Matechik, SM ja MR Stytz. 1994. “Using kriging to interpolate MRI data”. Teoksessa *Proceedings of 16th Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* 1:576–577. IEEE.

Microsoft. 2021. “File mapping”. Viitattu 1. marraskuuta 2023. https://learn.microsoft.com/en-us/windows/win32/memory/file-mapping.

———. 2022. “3D Graphics Overview”. Viitattu 1. marraskuuta 2023. https://learn.microsoft.com/en-us/dotnet/desktop/wpf/graphics-multimedia/3-d-graphics-overview.

Nielsen, Jakob. 1992. “Finding usability problems through heuristic evaluation”. Teoksessa *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 373–380.

———. 1994. “Ten usability heuristics”. Viitattu 1. marraskuuta 2023. https://www.nngroup.com/articles/ten-usability-heuristics/.

Nielsen, Jakob ja Thomas K Landauer. 1993. “A mathematical model of the finding of usability problems”. Teoksessa *Proceedings of the INTERACT’93 and CHI’93 Conference on Human Factors in Computing Systems,* 206–213.

Patro, S. Gopal ja Kishore Kumar Sahu. 2015. “Normalization: A preprocessing stage”. *International Advanced Research Journal in Science, Engineering and Technology* 2 (3). https://doi.org/10.17148/IARJSET.2015.2305.

Sayab, M., D. Aerden, J. Kuva ja W.U. Hassan. 2021. “Tectonic evolution of the Karakoram metamorphic complex (NW Himalayas) reflected in the 3D structures of spiral garnets: Insights from X-ray computed micro-tomography”. *Geoscience Frontiers* 12 (3). https://doi.org/10.1016/j.gsf.2020.11.010.

Schindelin, Johannes, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid ym. 2012. “Fiji: an open-source platform for biological-image analysis”. *Nature Methods* 9 (7): 676–682.

Shoemake, Ken. 1994. “Arcball Rotation Control”. Teoksessa *Graphics Gems IV,* toimittanut Paul S. Heckbert, 175–192. Academic Press.

Skodras, Athanassios, Charilaos Christopoulos ja Touradj Ebrahimi. 2001. “The JPEG 2000 still image compression standard”. *IEEE Signal Processing Magazine* 18 (5): 36–58.

Smith, Sidney L ja Jane N Mosier. 1986. *Guidelines for Designing User Interface Software.* Citeseer.

Spool, Jared ja Will Schroeder. 2001. “Testing web sites: Five users is nowhere near enough”. Teoksessa *CHI’01 Extended Abstracts on Human Factors in Computing Systems,* 285–286.

Thévenaz, Philippe, Thierry Blu ja Michael Unser. 2000. “Image interpolation and resampling”. *Handbook of Medical Imaging, Processing and Analysis* 1 (1): 393–420.

Wharton, Cathleen. 1994. “The cognitive walkthrough method: A practitioner’s guide”. Teoksessa *Usability Inspection Methods.* John Wiley.

Wiggins, Richard H, H Christian Davidson, H Ric Harnsberger, Jason R Lauman ja Patricia A Goede. 2001. “Image file formats: past, present, and future”. *Radiographics* 21 (3): 789–798.

Virzi, Robert A. 1990. “Streamlining the design process: Running fewer subjects”. Teoksessa *Proceedings of the Human Factors Society Annual Meeting,* 34:291–294. 4. SAGE Publications Sage CA: Los Angeles, CA.

———. 1992. “Refining the test phase of usability evaluation: How many subjects is enough?” *Human Factors* 34 (4): 457–468.

Vokolos, Filippos I ja Elaine J Weyuker. 1998. “Performance testing of software systems”. Teoksessa *Proceedings of the 1st International Workshop on Software and Performance,* 80–87.

Yang, Wuqiang, Zhen Ren, Masahiro Takei ja Jiafeng Yao. 2018. “Medical applications of electrical tomography”. Teoksessa *2018 IEEE International Conference on Imaging Systems and Techniques (IST),* 1–6. https://doi.org/10.1109/IST.2018.8577200.

Yoo, Byeongyeong, Jungheum Park, Sungsu Lim, Jewan Bang ja Sangjin Lee. 2012. “A study on multimedia file carving method”. *Multimedia Tools and Applications* 61:243–261.

Liitteet

A Poikkileikkauksien interpolointimenetelmien suorituskykyvertailun mittaustulokset.

Interpolointimenetelmä	Aika (s)
1. Lähin naapuri	93.611
2. Lähin naapuri	93.226
3. Lähin naapuri	93.385
1. Kolmilineaarinen	132.546
2. Kolmilineaarinen	132.621
3. Kolmilineaarinen	133.421
1. Kolmikuutio	128.712
2. Kolmikuutio	128.823
3. Kolmikuutio	128.765

Taulukko 4: Raakadatan suorituskykyvertailun mittaustulokset.

Interpolointimenetelmä	Aika (s)
1. Lähin naapuri	56.908
2. Lähin naapuri	60.973
3. Lähin naapuri	58.227
1. Kolmilineaarinen	110.018
2. Kolmilineaarinen	108.164
3. Kolmilineaarinen	111.771
1. Kolmikuutio	209.498
2. Kolmikuutio	210.505
3. Kolmikuutio	213.628

Taulukko 5: Kuvasekvenssin suorituskykyvertailun mittaustulokset.

Interpolointimenetelmä	Aika (s)
1. Lähin naapuri	69.851
2. Lähin naapuri	69.988
3. Lähin naapuri	70.586
1. Kolmilineaarinen	285.520
2. Kolmilineaarinen	291.066
3. Kolmilineaarinen	288.603
1. Kolmikuutio	1264.864
2. Kolmikuutio	1246.451
3. Kolmikuutio	1274.492

Taulukko 6: Raakadatan toisen suorituskykyvertailun mittaustulokset.