

Pistepilvien visualisointi laitossuunnitteluohjelmistossa

Pro Gradu
Turun yliopisto
Tulevaisuuden teknologioiden laitos
Tietojenkäsittelytiede
2018
Timo Heinonen
Tarkastajat:
P.P.
H.H.

Turun yliopiston laatujärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck-järjestelmällä

TURUN YLIOPISTO

Tulevaisuden teknologioiden laitos

Timo Heinonen Pistepilvien visualisointi laitossuunnitteluoohjelmistossa

Pro Gradu, 42 s., 3 liites.

Tietojenkäsittelytiede

14. joulukuuta 2019

Turun yliopiston laatuojärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check -järjestelmällä.

Juuhe elikkäs gradutyötä...

Sisältö

1 Johdanto	1
1.1 Pistepilvet	1
1.2 Laserkeilaimet	4
1.3 Pistepilvidatan käsittey	7
1.3.1 Panoramakuvat	7
1.3.2 Rekisteröinti	8
1.3.3 Muukalaispisteiden poistaminen	9
1.3.4 Pintojen rekonstruointi	10
1.4 Pistepilvien hyödyntäminen tietokoneavusteisessa suunnittelussa	10
1.5 Tutkielman tavoitteet ja rakenne	12
2 Pistepilvien käsitellyssä käytetyt tietorakenteet	14
2.1 Pistepilvien käsitellyn haasteet	14
2.2 Hierarkiset tietorakenteet	15
2.2.1 Qsplat	15
2.2.2 Peräkkäispistepuut	17
2.2.3 Sisäkkäispistepuut	18
3 Laitossuunnitteluoohjelmistoon optimoitu tietorakenne	23
3.1 Käyttötapaukset ja vaatimukset tietorakenteelle	23
3.1.1 Mallintaminen	23
3.1.2 Mittaaminen	24
3.1.3 Katselu	24
3.2 Tietorakenteen valinta	25
3.3 Pistedatan esitysmuoto	26
3.4 Tietorakenteen rakentaminen	30
3.5 Renderöinti	34

3.6 Pisteiden valinta	38
4 Tietorakenteen arviointi	39
4.1 Tietorakenteen rakentaminen ja lataaminen	39
4.2 Visualisointi	39
4.3 Pisteiden valitseminen	39
5 Johtopäätökset	40
Viitteet	40

1 Johdanto

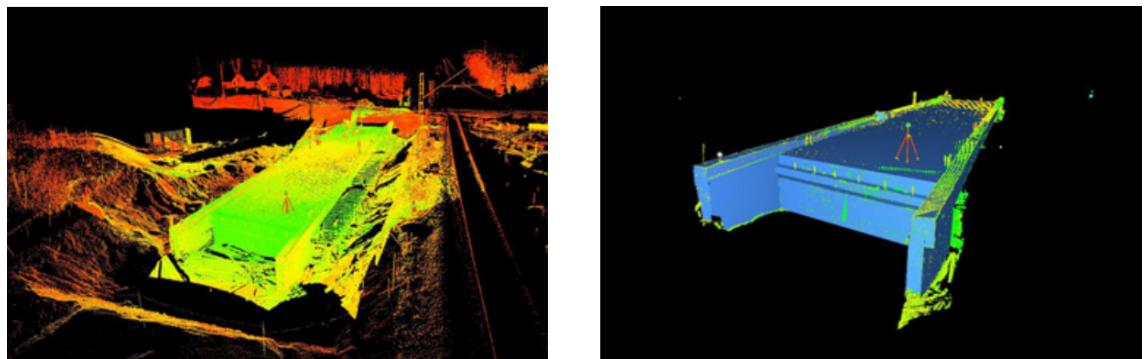
Tässä luvussa määritellään pistepilven käsite, tutustutaan laserkeilaimiin, joilla pistepilviä taltioidaan ja mainitaan muutamia yleisiä pistepilvien käsittelyn työvaiheita. Lisäksi esitellään muutama esimerkki pistepilvien käyttökohteista. Lopuksi määritellään tämän tutkielman tavoitteet, toteutustapa ja rakenne.

1.1 Pistepilvet

Pistepilveksi kutsutaan jotakin objektiota tai maisemaa kuvaavaa suurta joukkoa pisteitä kolmiulotteisessa avaruudessa. Pistepilvi tuotetaan yleensä laserkeilaimella (engl. *Laser Scanner*), joka ampuu ympärilleen laserpurskeita ja mittaa etäisyksiä pisteisiin, joista purske heijastuu takaisin. Pistepilviä voidaan tuottaa myös synteettisesti ottamalla näytepisteitä mistä tahansa 3d-mallista, mutta tässä tutkielmassa keskitytään laserkeilaimilla tuottuihin pistepilviin.¹

Pistepilville on useita käyttökohteita, joista arkkitehtuuri ja rakentaminen on yksi tärkeimmistä. Pistepilvien hyödyntäminen voidaan aloittaa rakennusprojektiin hyvin varhaisessa vaiheessa. Rakennettavan tontin ympäristöstä voidaan ottaa laserkeilausia, jotta suunniteltavan rakennuksen sopimista tontille voidaan helposti arvioida. Rakennusprojektiin aikana säädöllisillä laserkeilauskilla voidaan seurata tarkasti projektin etenemistä ja havaita mahdollisia ongelmia ajoissa. Pistepilvillä on myös tärkeä rooli rakennuksen valmistumisen jälkeen. Muutostöitä tehtäessä halutaan rakennuksesta saada ajantasalla oleva 3d-malli. Manuaalinen mallintaminen olisi hyvin suuri työ verrattuna muutaman kymmenen pistepilven luomiseen laserkeilaimella, mikä voidaan tehdä päivässä. [2]

¹1980-luvulta lähtien pisteitä on ehdotettu yleisiksi renderöintiprimitiiveiksi kuvaamaan mitä tahansa geometriaa [1]. Ajatus on nykyään vielä ajankohtaisempi, sillä renderöitäävät mallit monimutkaistuvat jatkuvasti ja usein yksittäiset polygonit projisoituvat kuvaruudulle alle pikselin kokoisina. Tällöin polygonien kärkipisteiden sijaan olisi tehokkaampaa säilyttää muistissa vain yhtä pistettä. Grafiikkakirjastot ja näytönohjaimet on kuitenkin vielä optimoitu kolmioiden käsittelyyn.



Kuva 1: Silta Joroisten ja Varkauden välissä Kuvasintiellä. Vasemmalla pistepilvi, oikealla pistepilvestä muodostetua geometriaa suunnitteluoohjelmassa. Kuva: [3]

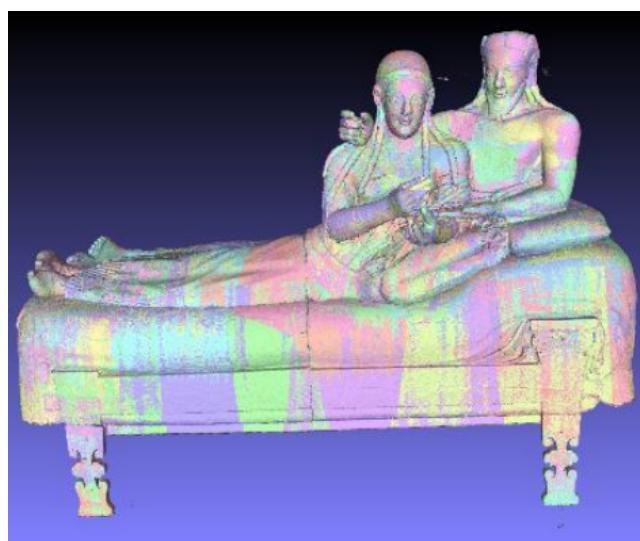
Eräs mielenkiintoinen sovelluskohde laserkeilaukselle on sillanrakentaminen. Sillanrakentamisessa haasteita tuottavat tien ja maaston geometrian yhteensovittamisen lisäksi projektin pitkä kesto. Silta on rakennettava osissa ja lopputuloksen on oltava suora, minkä takia siltatyömaalla suoritetaan usein tarkistusmittauksia. Älykäs silta -projektissa tutkittiin tapoja hyödyntää tietotekniikkaa sillanrakentamisessa ja -korjaamisessa, ja havaittiin laserkeilauksen olevan hyvä tapa suorittaa tarkkuusmittauksia. Laserkeilauksella muodostettiin pistepilviä sillan kannesta ja rakenteista, jonka jälkeen niistä muodostettiin pintoja 3d-suunnitteluoohjelmaan. Pistepilvi ja siitä muodostettu geometria on esitetty kuvassa 1. [3]

Pistepilviä käytetään hyväksi myös arkeologiassa. Roomalaiset perustivat vuoden 40 tienoilla Tonavan varrelle nykyisen Itävallan alueelle sotilasleirin, josta kasvoi myöhemmin Carnuntumin kaupunki. Kaupungin muurien ulkopuolelle rakennettiin amfiteatteri, johon mahtui 13000 katsojaa. Vuonna 2007 Ala-Itävallan osavaltion hallitus aloitti arkeologiset kaivaukset amfiteatterin alueella, joiden yhteydessä alueesta muodostettiin kattava pistepilvi noin kahdella sadalla laserkeilauksella. Ruudunkaappaus kysesestä pistepilvestä on esitetty kuvassa 2. [5]

Pistepilviä käytetään historiallisen kulttuuriperinnön säilyttämiseen myös pienemmäsä mittakaavassa. 1800-luvulla tehdyissä Cerveterin arkeologisissa kaivauksissa Italiassa löydettiin 500-luvulla terrakottasavesta tehty etruskilainen sarkofagi, joka kuvasi avio-



Kuva 2: Carnuntumin kaupungin amfiteatteri taltioituna laserkeilauksella. Kuva: [4]



Kuva 3: Etruskilaisesta sarkofagista muodostettu pistepilvi. Kuva: [6]

paria rentoutumassa tuonpuoleisessa. Satoihin palasiin hajonnut sarkofagi restauroitiin vuonna 1893 ja se digitoitiin käyttämällä laserkeilausta ja fotogrammetriaa vuonna 2013 taidenäytelyä varten. Sarkofagista keilattu pistepilvi on esitetty kuvassa 3. [6]

Eräs vaativa pistepilvien sovelluskohde on itseohjautuvat kulkuneuvot. Voidakseen navigoida liikenteessä itseohjautuva auto tarvitsee kameroiden ja ultraäänisensoreiden lisäksi katollen laserkeilaimen, jolla voidaan tarkkailla auton etäisyyttä muihin tienkäytäjiin ja esteisiin. Itseohjautuvat autot ovat merkittävä tutkimuskohde myös pistepilvien käsitelyn kannalta. Auton katolle asennettavan laserkeilaimen tulisi olla tarkka, nopea ja edullinen, ja sen tuottamaa pistedataa täytyy voida käsitellä reaalialjassa. [7]

Pistepilviä voidaan käyttää myös huomattavasti suuremmassa mittakaavassa. Maanmittauslaitos on kerännyt ilmasta käsin pistepilvidataa lähes koko Suomen maaperästä. Lentokoneesta keilattu pistepilvi on melko harva - puoli pistettä neliömetriä kohden -, mutta keilattavan kohteen laajuus tekee pilvistä valtavia. Pistepilviä on käytetty lähinnä metsävarojen kartoittamiseen, mutta Maanmittauslaitos aikoo ryhtyä keräämään pistedataa tarkemmillä laserkeilamilla myös rakennuksista. [8]

1.2 Laserkeilaimet

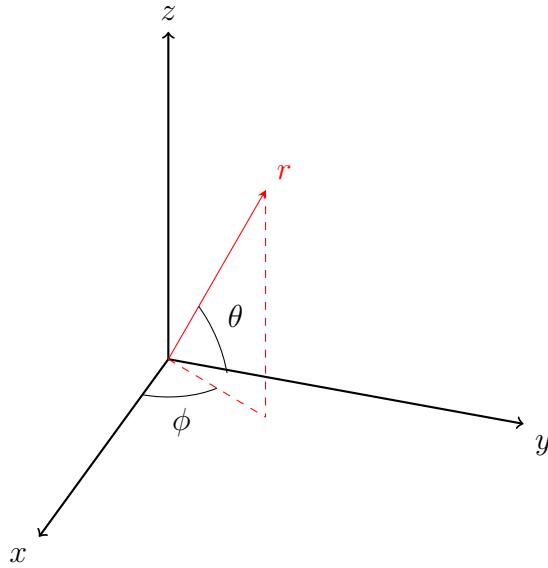
Perinteinen pistepilvien mittaanmiseen käytetty laserkeilain on jalustalla seisova laite, joka pyörii pystyakselinsa ympäri ampuen ympärilleen miljoonia laserpurskeita. Laserkeilain mittaa etäisyyksiä pisteisiin, joista laserpurske heijastuu takaisin keilaimeen ja muodostaa näistä pisteistä pistepilven. Heijastuksen voimakkuutta käytetään usein määräämään pistelle väri. Yleensä tarkasteltavaa kohdetta täytyy keilata useista eri suunnista, jotta saataisiin tarpeeksi kattava joukko pistepilviä. Kohteesta riippuen voidaan tarvita jopa satoja keilausia. Pistepilvien sovittamista yhteen tarkastellaan luvussa 1.3.

Nykyikaisella laserkeilaimella saadaan muodostettua hyvin tarkka ja tiheä pistepilvi nopeasti. Esimerkiksi kuvassa 4 näkyvä Leica Geosystems RTC360 -keilaimen luvataan mittaamaavan jopa kaksi miljoonaa pistettä sekunnissa ja kiertävän täyden ympyrän alle



Kuva 4: Kulkuaikateknikaan perustuva Leica RTC360 -laserkeilain.

Kuva: https://leica-geosystems.com/-/media/images/leicageosystems/about-us/news%20room/reporter/reporter-83/09-discovering-the-power-of-scanning/leica-espresso_expert_insights_640x750_slider4.ashx?la=en&hash=9D35592CBB571C3777E1E8A9A0A056BD



Kuva 5: Pallokoordinaatisto. Pisteen sijainti avaruudessa ilmaistaan korotuskulmalla θ , atsimuuttikulmalla ϕ ja säteellä r .

kahdessa minuutissa. Keilaimen lisäksi laitteessa on kamera, jolla saadaan määritettyä pisteille oikeat värit. [9]

Laserkeilaimien toimintaperiaatteissa on eroja. Kaksi yleisintä toimintaperiaatteita ovat kulkuaikateknikka (engl. *Time-Of-Flight*) ja vaihesiirtotekniikka (engl. *phase shift*). Kulkuaiateknikkassa pisteen etäisyys keilaimesta selviää ajasta, joka kuluu laserpurskeen lähetettämisestä sen heijastuksen vastaanottamiseen. Pisteen etäisyys keilaimesta on yksinkertaisesti

$$d = \frac{c \cdot t}{2}, \quad (1)$$

missä c on valonnopeus ja t on mitattu aika. [10] Vaihesiirtotekniikka perustuu keilaimesta lähtevän signaalin vaiheen vertaamista palaavan signaalin vaiheeseen. Pisteen etäisyys keilaimesta saadaan laskemalla

$$d = n \cdot \lambda + \frac{\Phi \cdot \lambda}{2 \cdot \pi}, \quad (2)$$

missä n on havainnon täysien aaltojen määrä, λ on signaalin aallonpituuus ja Φ on lähtevän ja palaavan signaalin vaihe-ero. [10]

Laserkeilain tallentaa mittaan pisteet pallokoordinaateissa. Pisteiden siirtäminen pallokoordinaateista karteesiseen koordinaatistoon onnistuu laskemalla koordinaatit

$$\begin{aligned}x &= r \cdot \sin \theta \cdot \cos \phi, \\y &= r \cdot \sin \theta \cdot \sin \phi, \\z &= r \cdot \cos \theta,\end{aligned}\tag{3}$$

missä r on pallon säde, θ on korotuskulma ja ϕ atsimuuttikulma. Pallokoordinaatistoa on havainnollistettu kuvassa 5.

1.3 Pistepilvidatan käsittely

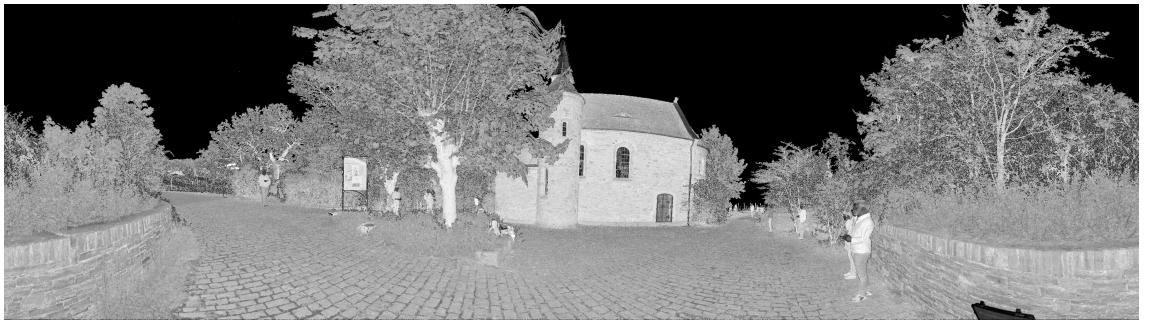
Laserkeilauksella tuotettuja pistepilviä ei usein käytetä sellaisenaan, vaan niitä täytyy ensin esikäsittellä. Yleisiä esikäsittelyvaiheita ovat panoramakuviien luominen, usean pistepilven rekisteröinti yhteen koordinaatistoon, muukalaispisteiden poistaminen ja pintojen rekonstruointi.

1.3.1 Panoramakuvat

Yksinkertaisin tapa visualisoida pistepilvi on muodostaa siitä kaksiulotteinen kuva. Pilvestä voidaan luoda panoramakuva projisoimalla laserkeilaimen ympäröimät pisteet kaksiulotteiselle kuvatasolle.² Kuvakoosta riippuen voidaan pistepilvestä karsia huomattava määrä pisteitä, joiden koko olisi liian pieni kuvatasolle projisoituna. Panoramakuvat toimivat siis myös pistedatan pakkausalgoritmina. Panoramakuvan pikseleille voidaan antaa värit joko sen perusteella, kuinka paljon valoa heijastuu takaisin niitä vastaavista pilven pisteistä, tai kuinka kaukana pisteet ovat keilaimesta. Panoramakuvaan on helppo soveltaa erilaisia kuvankäsittelyalgoritmeja, kuten piirteentunnistusta (engl. *feature detection*).

Kuvassa 6 on esimerkki panoramakuvasta.

²Projektiota vaikuttaa suuresti panoramakuvan laatuun. Hyvän yleiskatsauksen erilaisiin projektioihin antaa [11]. Kuvassa 6 on käytetty tasavälistä lieriöprojektiota (engl. *equiangular projection*).



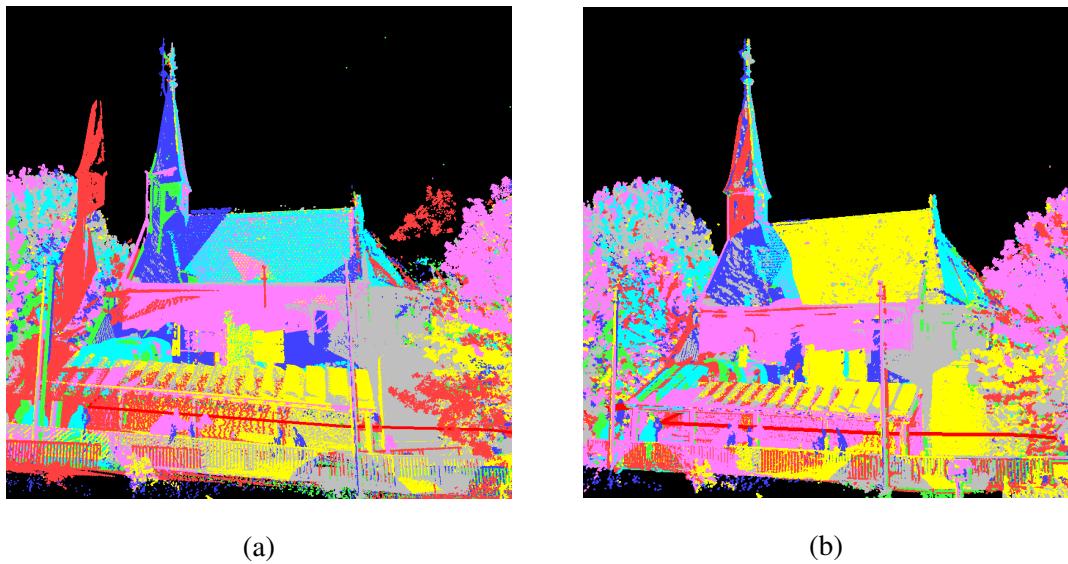
Kuva 6: Saksan Würzburgissa sijaitsevasta Randersackerin Neitsyt Marian kärsimysten kappelista keilatusta pistepilvestä muodostettu panoramakuva.

1.3.2 Rekisteröinti

Laserkeilauksen tuottama pistepilvi sisältää joukon pisteitä koordinaatistossa, jonka origona on keilaimen sijainti. Usein keilauksen kohteesta otetaan kymmeniä tai jopa satoja keilauksia, jotka täytyy saada samaan koordinaatistoon. Tätä kutsutaan pistepilven rekisteröinniksi.

Pistepilviä voidaan rekisteröidä usealla eri tavalla. Joskus keilattavaan kohteeseen asetetaan erityisiä merkkikuvioita, jotka näkyvät useasta keilaimesta. Kun tiedetään merkkien etäisyyys ja suunta kustakin keilaimesta, voidaan pistepilvet sovittaa helposti yhteen koordinaatistoon. Joissakin sovelluksissa käyttäjä merkitsee kahdesta pilvestä joukon pisteitä, joiden avulla pilvet saadaan rekisteröityä. Esimerkiksi 3DTK-kirjastolla käyttäjä voi valita kahdesta pilvestä samaa aluetta kuvaavia pisteitä, joiden avulla pilvien yhteensovitus onnistuu automaattisesti [12].

Pistepilviä voidaan rekisteröidä myös ilman merkkikuvioita tai käyttäjän apua. Iteratiivinen lähimmän pisteiden algoritmi (engl. *iterative closest point, ICP*) sovittaa pistepilven toiseen etsimällä rotaation ja translaation, jolla pilvien välinen virhe saadaan minimoitua. Virheen laskemista varten täytyy määrittää pilvistä toisiaan vastaavat pisteet. Yksinkertaisimillaan pistettä vastaavaksi pisteeksi valitaan toisesta pilvestä se, jonka euklidinen etäisyyys edellä mainittuun on pienin. Iteratiivisen algoritmin kunkin transformaation hyväys lasketaan kaikkien vastaavien pisteiden välisten etäisyyksien muodostamasta vir-



Kuva 7: Kuvan 6 kappelista keilaittuja pistepilviä (a) ennen rekisteröintiä ja (b) rekisteröinnin jälkeen. Kunkin keilaksen pisteet on piirretty eri väreillä.

heestä. Algoritmin iteroiminen voidaan lopettaa, kun jokin ennaltamääritetty raja virheelle on alitettu. Kuvassa 7 on ICP-algoritmilla sovitettu yhteen kuusi pistepilveä. [13]

ICP-algoritmi tarvitsee kuitenkin käyttäjältä hyvän alkuarvauksen, jotta virhe supensi ja pilvien sovittaminen onnistuisi. Se on myös erittäin raskas suorittaa isoille pistepilville. Yleensä ICP-algoritmia käytetäänkin karkeamman rekisteröintialgoritmin jälkeen hienosäätämiseen. Karkeampi pilven yhteensovittaminen tehdään yleensä etsimällä kahdesta pistepilvestä muodostetuista panoramakuivista toisiaan vastaavia avainpisteitä (engl. *keypoints*). Näiden avainpisteiden sijainneista saadaan harva joukko pisteitä, joiden yhteensovittaminen on paljon helpompaa kuin kokonaisten pistepilvien. Suosittuja teknikoita avainpisteiden löytämiseen ovat esimerkiksi Harris'n nurkat [14] sekä FAST ja SIFT -piirteet [15][16]. [17]

1.3.3 Muukalaispisteiden poistaminen

Pistepilvissä on usein muukalaispisteitä (engl. *outlier*), johtuen esimerkiksi laserkeilailmen epätarkkuudesta tai vaikkapa tuulen heiluttamista puiden lehdistä. Yksinkertainen

tekniikka poistaa muukalaispisteitä pilvestä on verrata pisteen normalivektoria sen naapuripisteiden normaaleihin. Pisteen p_i normalivektori voidaan selvittää pääkomponenttianalyysillä (engl. *principal component analysis, PCA*). Ensin on etsittävä pilvestä pisteen p_i k lähintä naapuria, jonka jälkeen naapuriston pisteistä lasketaan ominaisarvot. Kahda suurinta ominaisarvoa vastaavaa ominaisvektoria voidaan käyttää kuvaamaan tasoa, joka sovitetaan naapuruston päälle. Jäljelle jäävä ominaisvektori kuvailee pisteen p_i normaalialia. [18]

1.3.4 Pintojen rekonstruointi

Joissakin sovelluksissa halutaan luoda pistedatasta kohteen pintoja kuvaava polygoniverkko, jotta renderöinti olisi nopeampaa olemassaolevilla grafiikkakirjastoilla ja visuaalinen lopputulos parempi. Yksinkertainen tekniikka luoda tiivis kolmiointi pistepilvestä on Delaunayn kolmiointi³ (engl. *Delaunay triangulation*). Delaunayn kolmiointi perustuu Voronoin diagrammiin (engl. *Voronoi diagram*), joka jakaa pisteitä sisältävän tason tai avaruuden konvekseihin Voronoin soluihin. Voronoin solu kattaa sen alueen, jossa etäisyys solua vastaavaan pisteeseen on pienempi kuin muihin pisteisiin. Kahden solun välillä on Voronoin jana, josta etäisyys kahteen pisteeseen on sama, ja kolmen janan leikkauspisteessä on Voronoin kärki, josta etäisyys kolmeen pisteeseen on yhtä suuri. Delaunayn kolmiointi on Voronoin diagrammin duaaligraafi, josta luodaan graafi siten, että pisteen välillä on kaari, mikäli niitä vastaavat Voronoin solut jakavat Voronoin janan. [20]

1.4 Pistepilvien hyödyntäminen tietokoneavusteisessa suunnittelussa

Aiemmin mainittiin erilaisia sovelluksia laserkeilainten tuottamille pistepilville. Tässä tutkielmassa keskitytään pistepilvien hyödyntämiseen tietokoneavusteisessa suunnittelussa (engl. *computer aided design, CAD*) ja erityisesti laitossuunnittelohjelmistoissa (engl.

³Delaunayn kolmiointi sopii harvoin oikeilla laserkeilaimilla taltioituihin, häiriötä sisältäviin pistepilviin. Hyvän yleiskatsauksen realistisemmista pinnanmuodostustekniikoista ovat julkaisseet esim. [19]

plant design software).

Tietokoneavusteisessa suunnittelussa pistepilviä käytetään olemassaolevien rakenteiden taltiointiin. Usein käytetty esimerkki on autoteollisuuden alalta: ryhmä suunnittelijoita kokeilee uutta korimallia rakentamalla prototyppiauton helposti muovattavasta materiaalista. Kun prototyppi on todettu aerodynaamiseksi ja miellyttävän näköiseksi, täytyy se saada digitoitua, jotta se voidaan siirtää massatuotantoon. Usein yksinkertaisin ja kustannustehokkain tapa on laserkeilata prototyppi ja jatkokäsittellä pistepilveä niin, että saadaan luotua haluttu 3d-malli.

Laitossuunnittelussa yleisempi ongelma on 3d-mallin vanhentuminen tai sen puuttuminen kokonaan. Vaikka laitosta alunperin suunniteltaessa siitä olisi tehty 3d-malli, laitteistojen sommittelua saatetaan muuttaa ilman, että samoja muutoksia tehdään 3d-malliin. Kun 3d-mallia halutaan taas hyödyntää, voi olla kustannustehokkaampaa luoda laitoksesta laserkeilaimella pistepilvi kuin mallintaa tehdyt muutokset suunnitteluoohjelmalla. [21]

Kuten sillanrakennuksessa, myös laitoksia rakentaessa on joskus syytä suorittaa tarkeusmittauksia ja verrata niitä alkuperäiseen 3d-malliin. Pistepilvet ovat tähänkin tarkeuteen sopivia. Nykyaiosten laserkeilainten tuottamat pistepilvet ovat niin tarkkoja, että niistä voi havaita esimerkiksi putkien roikkumisen ja lämpölaajenemisen [21].

Pistepilviä voi hyödyntää monin tavoin laitossuunnittelussa. Jos laitokseen halutaan vaikkapa uusi vesiputki, voidaan sen mahtuminen varmistaa reitittämällä putki suunnitteluoohjelmistolla ja tarkastamalla, osuuko se pistepilveen. Jos taas vanhasta laitoksesta halutaan luoda ajantasalla oleva 3d-malli, voi suunnittelija mallintaa laitosta pistepilven avulla. Pistepilven päälle on helppo sijoittaa suunnitteluoohjelman putkistoja ja laitteita oikeille paikoilleen. Markkinoilla on myös ohjelmistoja, joiden luvataan tuottavan pistepilvestä automaattisesti älykäs 3d-malli komponenttitietoineen ilman aikaavievää päälle-mallinnusta [22].

Tietokoneavusteisen suunnittelun ohjelmisto käsittelee pistepilviä eri tavoin riippuen suunnittelutyön luonteesta. Kappalemallinnuksessa visualisoidaan usein yksittäisiä osia,

joita kuvaavat pistepilvet ovat hyvin tiiviitä ja usein kaikki pisteet ovat kerralla näkyvissä. Laitossuunnittelussa pistepilvet ovat taas hyvin laajoja ja voivat kuvata esimerkiksi koko-naisista tehdasalueita. Näissä sovelluksissa on tärkeää pistepilven nopea rajaaminen niin, että vain näkymän sisältävät pisteet renderöidään. Usein on tarpeen myös käyttää eri tarkkuksia pistepilven eri osille; kaukana katsojasta sijaitsevista pisteistä renderöidään vain osa [23]. Joitakin yleisiä käyttötapauksia pistepilvien kanssa työskentelystä laitossuunnitelussa ja niiden esittämiä vaatimuksia laitossuunnitteluoohjelmistolle on esitetty luvussa 3.1.

1.5 Tutkielman tavoitteet ja rakenne

Tämän tutkielman tarkoituksesta on tutustua pistepilvien käsittelyn ja visualisoinnin tuottamiin ongelmiin ja selvittää niihin ratkaisuvaihtoehtoja alan julkaisujen pohjalta. Selvitystyön pohjalta kehitetään tietorakenne erityisesti laitossuunnitteluoohjelmiston tarpeisiin. Lopuksi arvioidaan tietorakenteen suorituskykyä ja soveltuvuutta tehtävään käytäen verrokkina yksinkertaista ei-hierarkista tietorakennetta.

Tässä luvussa on perehdytty pistepilviin ja laserkeilaimiin yleisellä tasolla. Luvussa 2 tutustutaan alan kirjallisuuteen ja esitellään tietorakenteita, joiden ominaisuuksia voisi hyödyntää myös laitossuunnitteluoohjelmistossa. Luvussa 3 määritellään vaatimuksia, joita laitossuunnitteluoohjelmisto asettaa pistepilvien käsittelyyn ja visualointiin käytetylle tietorakenteelle. Tämän jälkeen ehdotetaan luvussa 2 esitelyihin julkaisuihin perustuva, laitossuunnitteluoohjelmistoon soveltuva tietorakenne. Tätä tietorakennetta verrataan yksinkertaiseen toteutukseen luvussa 4.

Tämä tutkielma on tehty CADMATIC Oy:n toimeksiantona. CADMATIC on suomalainen ohjelmistoyritys, joka kehittää tuotteita laivojen ja laitosten tietokoneavustiseen suunnittelun. CADMATIC on toiminut alalla 1980-luvulta lähtien ja sillä on asiakkainaan yli tuhat organisaatiota yli viidestäkymmenestä maasta [24]. Yrityksen pääkonttori on Turussa. Tutkielmassa kehitettävä tietorakennetta testataan CADMATIC Plant Mo-

deller -laitossuunnitteluoohjelmistossa, sekä eBrowser-mallinkatseluohjelmistossa.

2 Pistepilvien käsitellyssä käytetyt tietorakenteet

Tässä luvussa selvitetään, mitä haasteita pistepilvien käsitteily asettaa tietorakenteille ja visualisointialgoritmeille. Lisäksi tutustutaan haasteisiin vastaaviin hierarkisista tietorakenteista kirjoitettuihin julkaisuihin.

2.1 Pistepilvien käsitellyn haasteet

Suurin haaste pistepilvien käsitellyssä on niiden koko. Nykyaintelijainen laserkeilain, kuten luvussa 1.2 esitetty Leica Geosystemsin RTC360, tuottaa pistepilven, jossa on satoja miljoonia pisteitä. Kun tällaisella keilaimella tehdään useita keilausia, on pisteiden määrä valtava. Oletetaan esimerkiksi, että suressa projektissa käytetään pistepilviä, joissa on yhteensä miljardi pistettä. Kun koordinaatit tallennetaan kolmella nelitavuisella liukuluvuulla ja värit RGB-muodossa kolmella tavulla ja lisätään perään vielä yksi täytetavu, voidaan yksi pilven piste esittää 16:lla tavulla. Miljardin pisteen pilvi olisi siis kooltaan 16 gigatavua. Tämän kokoista pilveä ei haluta pitää kerralla keskusmuistissa, vaan pisteitä haetaan levyltä muistiin ulkoisen muistin algoritmeilla (engl. *out-of-core algorithm*).

Pisteiden määrän vuoksi ei ole realistista olettaa, että kaikki pisteet voitaisiin renderöidä reaalialjassa. Pisteiden määrää voidaan karsia harventamalla pilveä esimerkiksi jaka-malla pilvi säännöllisiin kuutioihin, niin sanottuihin vokseleihin, (engl. *volume element, voxel*), ja näyttämällä vain yksi piste kustakin vokselista. Pilven harventaminen kuitenkin aiheuttaa yksityiskohtien katoamista pilvestä, joten sitä täytyy käyttää sovelluskohteesta riippuen maltillisesti.

Usein on hyväksyttävä, ettei pistepilveä saada visualisoitua reaalialjassa. Interaktiivisessa ohjelmistossa on tärkeää kuitenkin pitää ruudunpäivitystaajuus tarpeeksi korkeana. Tällöin voidaan pistepilvestä piirtää ruudulle ensin karkea yleiskuva, jota tarkennetaan vähitellen, ellei käyttäjä keskeytä renderointiä esimerkiksi vaihtamalla kuvakulmaa. Tällainen astettainen visualisointi on mahdollista käyttämällä hierarkisia tietorakenteita pistepilven käsitellyssä. Tämän tekniikan etuna on se, että käyttäjä näkee välittömästi piste-

pilven yleisen muodon ilman, että hänen täytyy odottaa, että koko pilvi on visualisoitu. Jos käyttäjällä riittää kärsivällisyys, näkee hän kaikki pilven yksityiskohdat, kun renderöintialgoritmi on käynyt koko tietorakenteen läpi.

2.2 Hierarkiset tietorakenteet

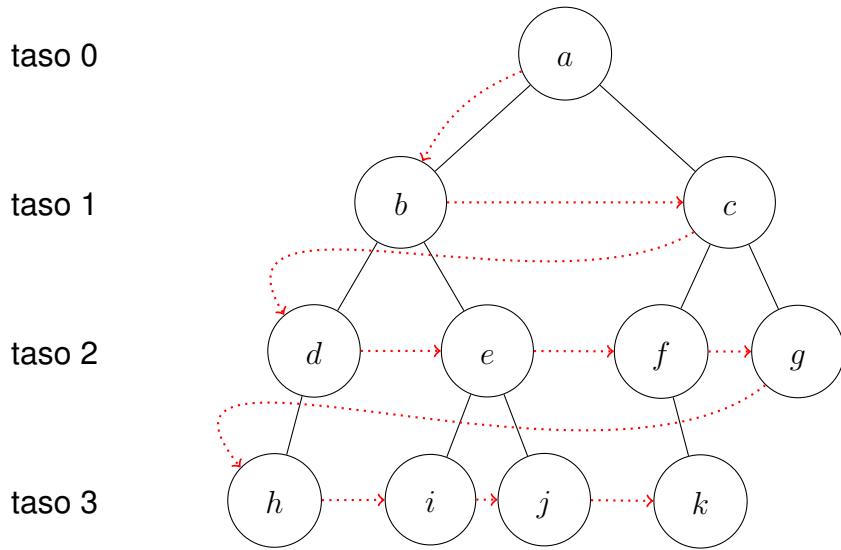
Pistepilvien käsittely on paljon tutkittu aihe ja siihen käytetyistä tietorakenteista löytyy paljon julkaisuja. Esitellään seuraavaksi muutama kiinnostava hierarkinen tietorakenne. Aihealueen pioneerityönä pidetään Rusinkiewiczin ja Levoyn Qsplatia, joka onkin antanut vaikutteita uudemmille tietorakenteille. Dachsbacher et al. esittelivät peräkkäispistepuut, jotka voidaan renderoidä hyvin nopeasti näytönohjaimella. Viime vuosina pistepilvien visualisoinnin tutkimuksen kirkkainta kärkeä on edustanut Wienin teknillisen yliopiston tietokonegrafiikan tutkimusyksikkö. Tämän tutkielman päälähteinä käytetään Claus Scheiblauerin ja Markus Schützin julkaisuja sisäkkäispistepuista.

2.2.1 Qsplat

Yksi ensimmäisistä pistedatan visualisointiin käytetyistä hierarkisista tietorakenteista on Rusinkiewiczin ja Levoyn esittämä QSplat, joka on kehitetty polygoniverkon visualisointiin pisteen avulla. Tietorakenne muodostetaan kolmiodusta mallista, jossa kolmioiden normaalit tunnetaan, joten se ei suoraan soveltu raa'an pistepilvidatan käsitteilyyn.⁴ QSplatissa on käytetty kuitenkin monia kiinnostavia tekniikoita, joita voi hyödyntää pistepilvien käsitteilyssä. [25]

QSplat perustuu puurakenteeseen, jonka solmuissa on rajauspalloja (engl. *bounding sphere*). Pallot jakavat avaruutta rekursiivisesti pienempiin osiin siten, että juuren pallo sisältää kaikki kolmiot ja jokainen sisäinen solmu jakaa avaruuden keskimäärin neljään osaan. Puun latva saavutetaan, kun avaruuden jakamisen seurauksena jäljelle jää yksi kol-

⁴Itse asiassa Rusinkiewicz ja Levoy käyttivät laserkeilatusta pistepilvestä muodostettua kolmioverkkoa, jonka tietorakenne esitti yksinkertaistettuna pistedatana.



Kuva 8: Puun läpikäyntijärjestys (punainen katkoviiva) muodostaa luonnolliset tarkkuus-tasot

mio. Siitä muodostetaan lehtisolmu, jonka rajauspallo sisältää koko kolmion. Puun visualisointi onnistuu piirtämällä jokaisen pallon kohdalle sopivan kokoinen täplä (engl. *splat*). Puurakenne mahdollistaa myös tehokkaan pisteiden karsimisen. Jos solmun pallo ei ole näkökentässä, eivät sen lapsetkaan ole ja haaraa ei tarvitse käydä läpi. [25]

Puurakenne tallennetaan levylle leveysjärjestyksessä (engl. *breadth-first*). Tämän ansiosta puun tasot muodostavat luonnolliset tarkkuustasot (engl. *level-of-detail, LOD*): juurisolmun pallo esittää koko mallia, ensimmäinen taso sisältää muutaman pienemmän pallon, ja niin edelleen. Kun tällainen tiedoston sisäinen rakenne yhdistetään ulkoisen muisin teknikoihin, voidaan täplien piirtäminen aloittaa heti, kun tarpeeksi puun solmuja on ladattu levyltä muistiin.⁵ Puun rakennetta on havainnollistettu kuvassa 8. [25]

Toinen hyödyllinen QSplatissa käytetty tekniikka on koordinaattien kvantisointi (engl. *quantization*). Kun tarkkuudesta voidaan tinkiä, solmujen pallojen absoluuttisia koordinaatteja ei tallenneta, vaan niiden sijainti ilmaistaan suhteessa vanhempiinsa. Pallon säteen ja keskipisteen suhteellisen poikkeaman ilmaisemiseen käytetään vain 13:a arvoa.

⁵Rusinkiewicz ja Levoy päättävät ruudulle projisoitujen täplien koon perusteella kuinka syvälle puussa tulee edetä.

Pallon sade r voi olla välillä $[\frac{1}{13}, \frac{13}{13}]$ ja samaten keskipisteen suhteellisen poikkeaman x, y ja z -koordinaatit ovat vanhemman pallon läpimitan kolmastoistaosan monikertoja. Kun vielä hylätään vanhemman pallon ulkopuolella olevat keskipisteet ja käytetään hakutauhua, voidaan pallon sijainti esittää vain 13:lla bitillä, kun normaali liukulukuesitys vaatisi vähintään 16 tavua. [25]

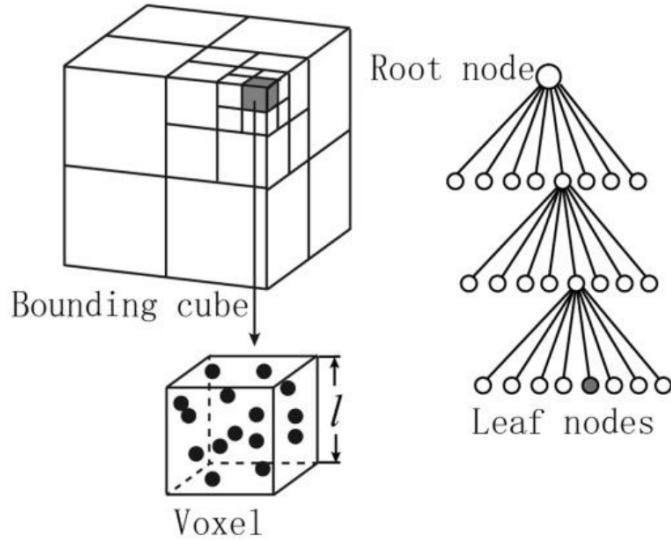
QSplat onnistui renderöimään 1,5-2,5 miljoonaa pistettä sekunnissa, mikä on sen aikaisella laitteistolla erinomainen tulos [25]. Kuten sanottu, se ei sellaisenaan kuitenkaan sovella laserkeilattujen pistepilvien käsittelyyn. Pistepilvien pisteiden normaaleja ei yleensä tiedetä, joten ne pitäisi esiprosessointivaiheessa selvittää esimerkiksi luvussa 1.3 esitettyllä tekniikalla. QSplat tarjoaa kuitenkin monia tekniikoita, joita pistepilviä käsittelvässä tietorakenteessa voidaan hyödyntää, kuten hierarknen rakenne ja koordinaattien suhteellinen esitystapa.

2.2.2 Peräkkäispistepuut

Dachsbacher et al. esittelevät niin kutstutun peräkkäispistepuun (engl. *sequential point tree*), jossa pisteet on aluksi järjestetty samaan tapaan pallopuuhun kuin QSplatissa. Pisteet sijaitsevat puun lehtisolmuissa ja sisäsolmuissa säilytetään täpliä, jotka juuri ja juuri peittävät solmun lasten rajauspallot. Täplien väri määräytyy lapsisolmujen värien keskiarvolla. [26]

Jokaiseen puun solmuun on virhelaskelmien perusteella lisätty rajat katseluetäisyydelle, jolla solmu valitaan visualisoitavaksi. Visualisointivaiheessa hierarkia litistetään taulukoksi, joka voidaan syöttää suoraan näytönohjaimelle. Näytönohjain käy taulukkoa läpi ja valikoi sopivat solmut, joiden perusteella täpliä piirretään ruudulle. [26]

Peräkkäispistepuut voidaan renderoidä nopeasti näytönohjaimen käytön ansiosta, mutta niissä on myös heikkouksia. Tietorakenteen vaatimuksena on, että kaikki data mahtuu näytönohjaimen muistiin. Näin on vain pienillä malleilla ja tilannetta pahentaa se, että peräkkäispistepuut eivät ole kovin säästäväisiä muistin suhteen. Hierarkian jokaisessa si-



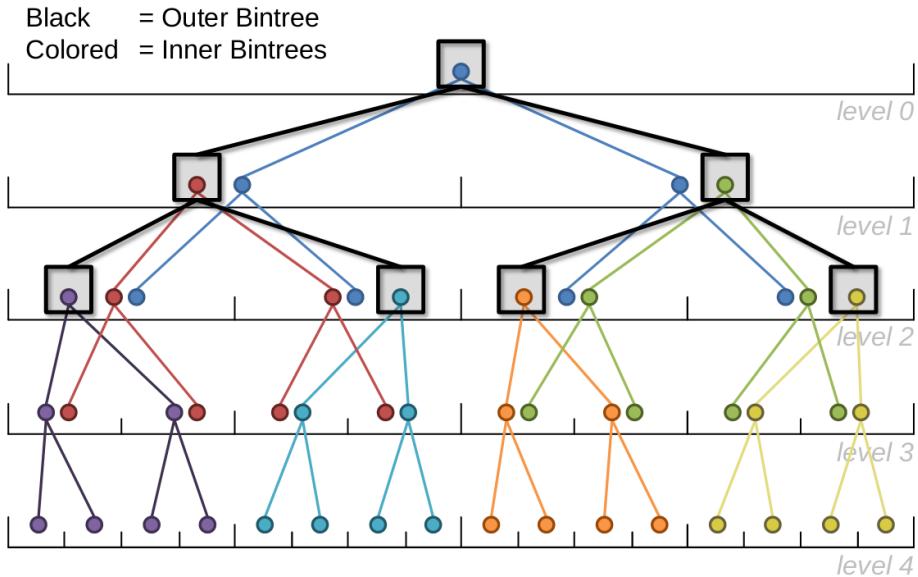
Kuva 9: Oktettipuun juurisolmu sisältää kaikki pisteet sisältävän rajauslaatikon. Jokainen sisäsolmu jakaa rajauslaatikkonsa kahdeksaan osaan. Kuva: [28]

säsolmussa luodaan lisää dataa, kun lapsisolmujen unionia kuvataan täplän sijainnilla ja koolla, sekä keskimääräisellä väriillä.

Wimmer ja Scheiblauer esittävät parannuksia peräkkäispistepuihin. Uusien täplien luomisen sijaan puun sisäsolmuissa valitaan lapsisolmuista edustaja, joka parhaiten kuvaa sisäolmun esittämää avaruuden osaa. Tämä on tärkeä huomio, sillä käsiteltäessä massiivisia pistepilviä tulisi välttää ylimääräisen datan luomista. Hierarkiasta muodostetaan tarkkuustasot siten, että alempat tarkkuustasot sisältyvät ylempien tasoihin ja visualisoitaessa oikea tarkkuustaso valitaan täplien koon ja katseluetäisyyden perusteella. Sollu vastaavan täplän koko määräytyy siitä, kuinka syvällä hierarkiassa se on. Wimmer ja Scheiblauer kutsuvat tästä rakennetta muistioptimoiduksi peräkkäispistepuksi. (engl. *memory optimized sequential point tree, MOSPT*) [27]

2.2.3 Sisäkkäispistepuut

Wimmer ja Scheiblauer kritisoivat muistioptimoitujen peräkkäispistepuiden siitä, että ne eivät tue näkökentän ulkopuolisten pisteiden tehokasta karsimista ja siitä, ettei muistiopti-



Kuva 10: Sisäkkäinen binääripuu, jossa sekä ulomman, että sisempien puiden syvys on kolme. Ulompaa puuta kuvaavat mustat neliöt ja sisempää värikkääät ympyrät. Puista muodostuu viisi tarkkuustasoa. Kuva: [30]

mointi yksinään riittänyt poistamaan tarvetta ulkoisen muistin algoritmeille. Ratkaisuksi he esittivät sisäkkäisiä oktettipuita (engl. *nested octree*). Oktettipuu⁶ on yksinkertainen avaruutta rekursiivisesti jakava tietorakenne, jonka jokainen sisäsolmu jakaa kuvaamansa avaruuden osan kahdeksaan osaan. Oktettipuun rakennetta on havainnollistettu kuvassa **??**. Wimmerin ja Scheiblauerin tietorakenteessa oktettipuita on kahdessa tasossa. Ulomppaa oktettipuuta käytetään avaruuden jakamiseen, sen tehokkaiseen läpikäymiseen ja näkökentän ulkopuolisten alueiden karsimiseen. Ulomman puun jokainen solmu sisältää yhden sisemmän oktettipuun, joka vastaa samaa avaruuden osaa, kuin ulkoisen puun solmu. Pisteet sijoitetaan sisempiin puihin, yksi jokaiseen solmuun. [27]

Sisäkkäisistä oktettipuista luodaan tarkkuustasot siten, että sisemistä puista kerätään pisteitä ulomman puun tasojen mukaan. Tarkkuustasoon kuuluvat pisteet sijaitsevat siis ulomman puun samalla tasolla, mutta useiden sisempien puiden eri tasolla. Tarkkuustasojen muodostumista on havainnollistettu kuvassa **??**. Puut tallennetaan levylle tarkkuus-

⁶Tätä suomennosta käyttää esimerkiksi Davidsson [29]. Vaihtoehtoinen suomennos on kahdeksanpuu.

taso kerrallaan, mikä mahdollistaa ulkoisen muiston algoritmien käytön. Visualisoitaessa tarvitsee levyltä lukea pisteitä vain haluttuun tarkkuustasoon asti, eikä loppuja pisteitä tarvitse ladata muistiin. [27]

Scheiblauer jalostaa sisäkkäisten oktettipuiden ideaa väitöskirjassaan esittelemällä muokkavat sisäkkäiset oktettipuut (engl. *modifiable nested octree, MNO*). Jos edellä esitellyjä sisäkkäisiä oktettipuita halutaan muokata rakentamisen jälkeen, on sisemmät puut rakennettava ja muokattu hierarkia tallennettava levylle uudestaan. Nimensä mukaisesti MNO mahdollistaa tehokkaan pisteiden lisäämisen ja poistamisen. [30]

MNO:n rakenne eroaa sisäkkäisistä oktettipuista siten, että sisemmät puut korvataan säädöllisillä, kolmiulotteisilla ruudukoilla, joihin pisteet tallennetaan. MNO:n rakentaminen alkaa juurisolmusta, joka vastaa kaikki pisteet peittävää avaruutta. Solmun sisältämä ruudukko jakaa solmua kuvaavan avaruuden osan $128^3 = 2097152$ soluun. Pisteitä lisätään puuhun yksi kerrallaan niin, että jokaiseen ruudukon soluun mahtuu vain yksi piste. Jos solu on varattu, sijoitetaan piste ylimääräiseen taulukkoon odottamaan, että vastaavia pisteitä kertyy tarpeaksi, jotta olisi järkevä luoda uusia solmuja puuhun. Kun ennaltaarattu vähimmäismäärä pisteitä on kertynyt ylimääräisten pisteiden taulukkoon, luodaan ruudukon sisältävälle solulle lapsisolmuja ja sijoitetaan ylimääräiset pisteet niihin. Ruudukkoon sijoitettavien pisteiden määälle on hyvä asettaa myös yläraja. [30]

Jokainen tietorakenteen solmu tallennetaan omaan tiedostoonsa levylle, josta niitä ladataan muistiin visualisointivaiheessa tarvittaessa. renderöintialgoritmien kuuluu käyttäjän asettama pistebudjetti, joka asettaa ylärajan yhdessä ruudunpäivityksessä piirrettävien pisteiden määälle.⁷ Tätä rajaa säättämällä käyttäjä saa jonkinlaisen kontrollin ruudunpäivitystaajuuden suhteeseen. [30]

Tiedostorakenne mahdollistaa hierarkian yksinkertaisen muokkaamisen. Lisättäessä uusia pisteitä MNO:hon tarkastetaan ensin, sijoittuuko se juurisolmun kuvaamaan avaruuden osaan. Jos näin on, onnistuu lisääminen kuten rakennusvaiheessa. Muussa tapauk-

⁷Scheiblauer testasi pistepilvivisualisoijaansa asettamalla rajan vain sataantuhanteen pisteeseen.

sessa juurisolmulle luodaan vanhempia kunnes jokin niistä muodostaa tarvittavan kokoinen avaruuden, ja piste lisätään sen ruudukkoon. Kun puun vanhan juuren yläpuolelle luodaan uusia solmuja, jää niiden ruudukot vajaaksi. Tällöin alemmista solmuista nostetaan pisteytä ylöspäin niin kauan, kunnes vajaita ruudukoita on vain lehtisolmuissa. Pisteiden poistaminen puusta on triviaalia, kun sisäsolmuihin mahdollisesti jäävät tyhjät ruudukot täytetään kuten pisteytä lisättäessä. [30]

Markus Schütz jatkoi Wimmerin ja Scheiblauerin työtä esittelemällä opinnäytetyösään verkkoselaimessa ajettavan Potree-nimisen pistepilvivisualisoijan. Potreen käyttämä tietorakenne perustuu Scheiblauerin muokattaviin sisäkkäisiin oktettipuihin, mutta hierarkian rakennusvaiheessa kiinnitetään huomiota pisteiden tasaiseen jakautumiseen solmujen välille. Oktettipuun sisäsolmujen ruudukoihin hyväksytään uusia pisteytä vain, jos ne ovat tarpeeksi kaukana muista ruudukon pisteytä. Lehtisolmut hyväksyvät ennaltamäärittyyn rajaan saakka kaikki pisteytä, kunnes ne muutetaan sisäsolmuiksi ja liian lähekkäin olevat pisteytä jaetaan uusien lapsisolmujen kesken. [31]

Potree käyttää ulkoista muistia tehokkaasti ja pystyy käsittelemään jopa 640 miljardia pistettä sisältäviä pistepilviä.⁸ Rakennusvaiheessa oktettipuun solmuja tallennetaan tasaisin väliajoin levylle, jottei muisti täyttyisi. Kun jokainen solmu tallennetaan omaan tiedostoonsa, on yksittäisten solmujen tallentaminen ja lukeminen levyltä helppoa. Massiivisia pistepilviä kuvaavat hierarkiatkin voivat olla satojen megatavujen kokoisia. Schütz ratkaisee suurten hierarkioiden nopean lataamisen verkon yli jakamalla senkin puurakenteeseen. Nämä voidaan välittää sekä turhien pisteytä, että näkökentän ulkopuolella olevien hierarkian haarojen lataaminen muistiin. [31]

Potreen visualisointialgoritmi priorisoii niitä hierarkian solmuja, jotka ovat lähellä kat-selupistettä ja joiden kuvaruudulle projisoitu koko on suurin. Renderöinnin suorituskykyä voidaan säädellä Scheiblauerin toteutuksen mukaisesti käyttäjän asettamalla pistebudje-

⁸Kyseinen pistepilvi (Actueel Hoogtebestand Nederland, ANH2, <http://ahn2.pointclouds.nl/>) kuvailee Alankomaiden valtiota ja se vaatii 7,68 teratavua tallennustilaan. Potreen tietorakenteessa pistepilvi jakautuu 13:lle tasolle ja 38:aan miljoonaan solmuun.

tilla. Schütz on kehittänyt Potreehen myös hienostuneen, näytönohjaimella ajettavan algoritmien mukautuvaan pisteiden koon määrittämiseen; pistepilven harvemmissa osissa piirretään pisteet suurempina, jottei reikiä esiintyisi. [31]

3 Laitossuunnitteluoohjelmistoon optimoitu tietorakenne

Tässä luvussa esitellään pistepilvien käsittelyyn ja visualisointiin soveltuva tietorakenne ja algoritmeja erityisesti laitossuunnitteluoohjelmiston tarpeisiin. Ensin määritellään vaatimukset tietorakenteelle tyypillisten pistepilvien käyttötapausten mukaan, jonka jälkeen valitaan alan kirjallisuudesta sovelluskohteelle hyödyllisimmät tekniikat.

3.1 Käyttötapaukset ja vaatimukset tietorakenteelle

Kolme yleistä käyttötapausta pistepilvien kanssa työskenneltäessä ovat mallintaminen, mittaaminen ja katselu, jotka asettavat erilaisia vaatimuksia pistepilviä käsitlevälle ja visualisoivalle laitossuunnitteluoohjelmistolle. Esitellään seuraavaksi käyttötapaukset ja niiiden asettamat vaatimukset.

3.1.1 Mallintaminen

Kun laitoksesta halutaan luoda ajantasalla oleva 3d-malli pistepilven avulla, täytyy se mallintaa suunnitteluoohjelmiston käyttämäksi geometriaksi pistepilveä mukailen. Lattiat ja seinät on tasoina helppo asettaa paikalleen, kuten myös suunnitteluoohjelmiston komponenttikirjastosta löytyvät laitteet. Suurin työ on yleensä putkistoissa, ilmakanavissa ja kaapeliradoissa. Useat suunnitteluoohjelmistot tarjoavat jonkinasteista automatisointia etenkin putkien reittykseen pistepilven päälle. Ohjelmisto voi automaattisesti tunnistaa pilvestä sylinterit ja asettaa niiden päälle sopivia putkisto-osia. Vaihtoehtoisesti käyttäjä voi valita pilvestä muutamia pisteitä ja ohjelmisto laskee niiden perusteella putken pituuden ja halkaisijan ja asettaa oikean osan paikalleen. Mallinnustyö ja etenkin automaattiset mudontunnistusalgoritmit asettavat ohjelmistolle vaatimuksen tarkkuudesta. Laitossuunnitteluoohjelmistossa käytetään yleensä millimetrejä perusyksikköinä, joten pistepilvessä ei sisisi esiintyä senttimetriien virheitä.

Mallintamisessa tärkeässä roolissa on suunnittelijan käyttämät näkymät ja pistepiven rajaaminen. Yleensä suunnittelija käyttää muutamaa koordinaattiakselien suuntaista nä-

kymää samanaikaisesti, jotta kurSORin saa helposti oIKEAan PAIKKAAN. NÄKYMÄN SYVYYS asetetaan usein hyvin pieneksi, jotta mallista näkyisi vain kulloisenkin mallinnustyön vaATIMA pieni siivu. Myös pistepilveä voidaan rajata niin, että siitä näkyy vain tarpeellinen osa. Pistepilviä visualisoivan ohjelmiston tulisi siis kyETÄ rajaamaan pilveä TOISTUVASTI ja NOPEASTI. KÄYTTÖKOKEMUS olisi paras, jos käyttäjä pystyisi hiirellä interaktiivisesti mÄÄRITTÄMÄÄN tilan, jonka sisäpuolella olevat pisteet renderöiTÄISIIN. Lisäksi pistepilvi tulee voida renderöidÄ useaan eri näkymään samanaikaisesti.

3.1.2 Mittaaminen

Toinen tärkeä ominaisuus pistepilvien kanssa työskennellessä on mittaaminen. Pistepilviä käytetään usein tarkastamaan, mahtuuko laitokseen jokin uusi laite tai putkisto. Tällöin on hyödyllistä suorittaa mittauksia joko kahden pistepilven pisteen, tai pisteen ja 3d-mallin geometrian välillä. Mittausoperaatiossa käyttäjä valitsee pistepilvestä kurSORilla haluamansa pisteen ja ohjelmisto palauttaa lähimäksi kursoria projisoidun pisteen. Käyttäjän kannalta olisi miellyttävää, jos mittausoperaatioita tehtäessä ei tarvitsisi odotaa, kun pistepilven miljoonien pisteiden joukosta etsitään juuri kurSORin alla oleva piste. Yksittäisten pisteiden hakeminen pilvestä tätyy siis olla nopeaa.

3.1.3 Katselu

Kolmas yleinen pistepilvien käyttökohde on 3d-mallin katselu joko laitossuunnitteluoHjelmistossa tai erityisessä mallinkatseluohjelmistossa. Etenkin suunnitteluprojektien esimiehet haluavat usein tarkastella suunnittelijoiden luomaa 3d-mallia helposti ja nopeasti. Luonnollisesti malliin kuuluvat pistepilvet tulevat myös näkyä katselijalle. Tämä saattaa tuottaa haasteita ohjelmiston kannalta, sillä katseluohjelmistojen käyttäjillä on käytettävissä harvoin yhtä järeää laitteistoa, kuin suunnittelijoiden työasemat. Mallinkatseluohjelmistossa pistepilveä harvemmin rajataan pienemmäksi, joten renderöiTÄVIÄ pisteitä on niin paljon, etteivät ne mahdu kerralla keskusmuistiin tai näytönohjaimen muistiin. Yleen-

sä käyttäjä myös liikuttaa näkymää mallin ympäri enemmän kuin mallinnustyössä, joten pistepilvisualisoinnin suorituskyky ja tarkkuustasot ovat entistäkin tärkeämpiä.

Tässä tutkielmassa kehitetään laitossuunnitteluoohjelmistolle optimoitu hierarkinen tietorakenne pistepilvien käsittelyyn. Esitetään tietorakenteelle seuraavat vaatimukset edellä mainittujen käyttötapausten perusteella:

1. On voitava visualisoida karkea yleiskuva pistepilvestä vain pienellä osalla datasta.
2. On käytettävä ulkoisen muistin algoritmeja, eli koko pilveä ei pidetä kerralla keskusmuistissa.
3. Pistepilven vaatima tallennustilan määrästä voidaan laskea harventamalla sen tiheasti näytteistettyjä osia.
4. Käyttäjän on voitava määrittää pilvestä alueita, joiden sisältävien tai ulkopuolelle jäävien pisteiden ominaisuuksia, kuten näkyvyyttä tai väriä, voidaan muuttaa.
5. Pilvestä on voitava nopeasti ja tarkasti valita yksittäisiä pisteitä.
6. Pistepilvessä ei saa esiintyä yli millimetrin suuruisia virheitä.

3.2 Tietorakenteen valinta

Luvussa 2.2 esitelly Potree on osoittaunut massiivisten pistepilvien interaktiivisen visualisoinnin olevan mahdollista jopa verkkoselaimessa, jossa etenkin tiedonsiirtonopeus rajoittaa renderöinnin nopeutta. Tarkastellaan siis sisäkkäispistepuiden soveltuvuutta laitossuunnitteluoohjelmistoon.

Oktettipuun läpikäyminen taso kerrallaan muodostaa tehokkaasti tarkkuustasot, joten vaatimus 1 on helppo tyydyttää. Pisteiden asettelu sisäkkäisten oktettipuiden solmuihin mahdollistaa myös vaatimuksen 2 mukaisesti ulkoisen muistin käyttämisen. Scheiblaueerin muokattavien sisäkkäisten oktettipuiden jokainen solmu sisältää ruudukon, johon pisteeet sijoitetaan. Mitä syvemmällä tasolla solmu on, sitä pienempiä ruudukon solut ovat.

Vaatimuksen 3 esittämä pilven harvennus onnistuu asettamalla puulle enimmäissyyvyys ruudukon koon mukaan ja hylkäämällä lehtisolmuissa kaikki pisteet, jotka tulisi lisätynsi jo varattuun soluun.

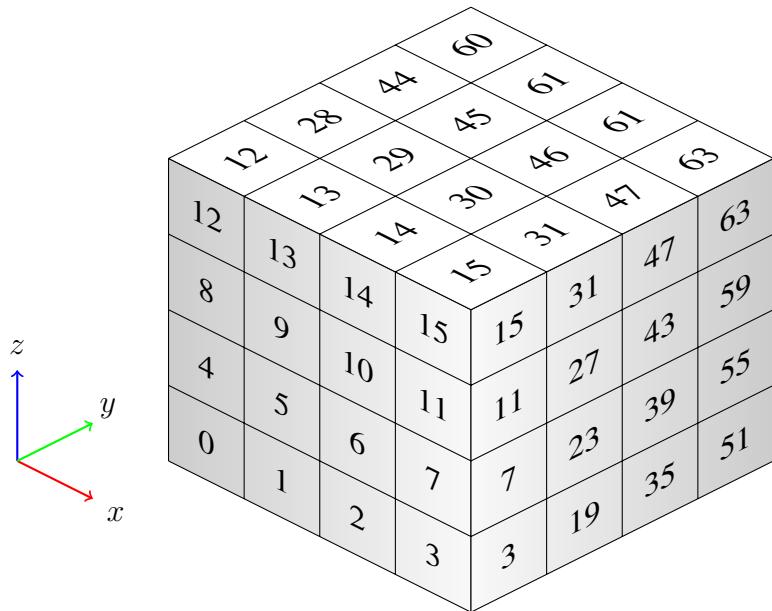
Valintaoperaatiot onnistuvat nopeasti oktettipuussa. Puun jokainen solmu sisältää tiedon sen sisältämien pisteiden rajauslaatikosta (engl. *bounding box*), joten jos valinnan sijainti ei osu rajauslaatikon sisälle, ei kyseisen solmun lapsisolmujakaan tarvitse tarkastaa. Yksittäisiä pisteitä tarvitsee tarkastella vasta kun valittavan alueen raja kulkee puun solmun rajauslaatikon läpi, tai kun käyttäjä haluaa valita vain yhden pisteen. Tietyllä aluella sijaitsevat pisteet jakautuvat useaan oktettipuun solmuun, minkä johdosta valintaoperaatiot eivät ole triviaaleja sisäkkäisissä oktettipuissa. Vaatimuksiin 4 ja 5 voidaan kuitenkin vastata sisäkkäisillä oktettipuilla. Scheiblauer ja Schütz eivät tiivistäneet pistepilviä, joten niiden tarkkuus ei kärsinyt. Näin myöskaän vaatimus 6 ei tuota ongelmia.

Sisäkkäispistepuut näyttävät siis soveltuvan myös laitossuunnitteluhjelmiston tarpeisiin. Scheiblauerin ja Schützin tietorakenteiden käyttämät ruudukot mahdollistavat kuitenkin myös joitakin laitossuunnittelusovelluksille tärkeitä optiomointeja. Esitellään seuraavaksi, miten pistedataa voi tiivistää ruudukon avulla.

3.3 Pistedatan esitysmuoto

Pistepilvet sisältävät usein satoja miljoonia tai jopa miljardeja pisteitä, mikä johtaa luonnollisesti isoihin tiedostokokoihin. Yleensä hierarkiatiedon osuus tiedoston sisällöstä on varsin pieni, joten tiedostokokoa saadaan pienennettyä parhaiten tiivistämällä itse pisteiden esitysmuotoa. Pisteistä tarvitsee tallentaa vähintään niiden sijainti ja väri. Yleensä sijainti esitetään kolmella nelitavuisella liukuluvulla ja väri RGB-koodauksella kolmella yksitavuisella kokonaisluvulla:

```
struct Point {
    float x;
    float y;
```



Kuva 11: 64:n solun ruudukko, jonka indeksointi alkaa vasemmasta alakulmasta

```
float z;  
  
unsigned char r;  
  
unsigned char g;  
  
unsigned char b;  
  
}
```

Yleensä näiden 15:a tavun lisäksi lisätään yksi pakkaustavu, jotta koko olisi mukava kahden potenssi. Miljardi pistettä tallennettuna tässä esitysmuodossa vaatii siis 16 gigatavua muistia.⁹ Muuttamalla pisteiden esitysmuotoa voidaan pistepilviä tiivistää ja joitakin operaatioita nopeuttaa.

Sisäkkäispistepuiden käyttämä ruudukko mahdolistaan yksinkertaisen pakkausalgoritmin. Puun rakennusvaiheessa pisteet lisätään jokaisessa solmussa olevaan kolmiulotteiseen ruudukkoon, jonka jokaiseen soluun mahtuu vain yksi piste. Kun piste lisätään ruudukon soluun, tallennetaan solun järjestysnumero hajautustauluun, josta voidaan jatkossa nopeasti tarkastaa, onko kyseinen solu varattu. Mahdollinen numeroointi ruudukolle on

⁹Pistepilvisovelluksen käyttäjän rahapussin koosta riippuen tämän kokoinen pilvi mahtuisi vielä keskusmuistiin ja jopa näytönohjaimen muistiin [32].

esitetti kuvassa 11. Numerointi alkaa vasemmasta alakulmasta indeksistä nolla ja etenee vasenkätisen koordinaatiston mukaisesti.

Puun solmuihin on tallennettu rajauslaatikko, joka määrittää myös ruudukon mitat ja sijainnin. Ruudukkoon lisättävien pisteen absoluuttinen sijainti voidaan unohtaa ja käyttää sijainnin tallentamiseen pisteen suhteellista sijaintia ruudukossa. Yksinkertaisimillaan voidaan kustakin pisteestä tallentaa vain sen solun indeksi, jossa piste sijaitsee. Tällöin pisteen esitysmuoto on siis

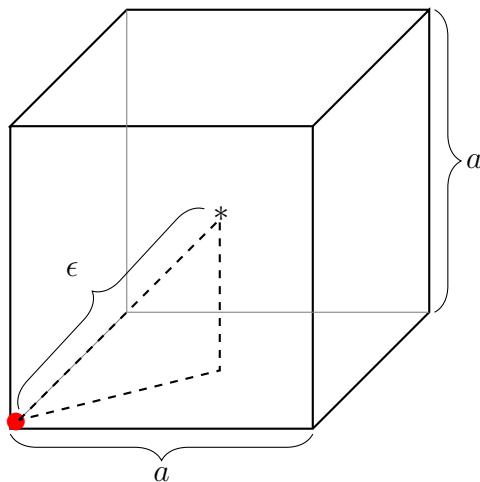
```
struct Point {
    unsigned int index;
    unsigned char r;
    unsigned char g;
    unsigned char b;
}
```

Kun solun indeksi tallennetaan etumerkittömänä nelitavuisena kokonaislukuna ja lisätään loppuun yksi pakkaustavu, tiivistyy piste kahdeksantavuiseksi. Näin miljardi pistettä välttisi enää 8 gigatavua muistia.

Esitysmuota voidaan tiivistää vielä tästäkin. Pisteen etäisyyttä suhteessa ruudukkoon voidaan esittää kolmella tavulla niin, että jokainen tavu kuvailee koordinaattiakselien suuntaisten askelten määrää ruudusta 0 lähtien. Yhden tavun esittämä enimmäisarvo on 255, joten ruudukossa voi olla enintään $255^3 = 16581375$ solua. Tällainen kompressio tuo toki pistepilveen epätarkkuutta, johon palataan myöhemmin.

Usein laitossuunnittelussa käytetyissä laserkeilaimissa ei käytetä värikameraa antamaan pisteille värejä, vaan väri-informaatio johdetaan keilaimeen heijastuvan valon määristä. Tämä arvo normalisoidaan yleensä välille $[0, 255]$, josta saadaan jokin harmaan sävy. Tällaisissa pistepilvissä on siis turha tallentaa jokaiselle pisteelle r , g ja b -arvoja, kun vain yksi arvo riittäisi. Näin piste voidaan esittää muodossa

```
struct Point {
```



Kuva 12: Suurin mahdollinen ruudukossa esiintyvä virhe ϵ . Kuutio kuvaaa ruudukon solua ja asteriski sen visualisointiin käytettävää keskipistettä. Soluun lisätty punainen piste on juuri ja juuri solun sisällä.

```

unsigned char dx;
unsigned char dy;
unsigned char dz;
unsigned char intensity;
}
```

eli tarvitaan vain neljä tavua tilaa ja edellä mainittu pistepilvi saadaan tivistettyä neljännekkseen alkuperäisestä koostaan.

Kun pisteen tarkka sijainti unohdetaan ja se ilmaistaan suhteessa ruudukkoon, syntyy pistepilveen virheitä. Suurinta mahdollista virhettä on havainnollistettu kuvassa 12, kun piste visualisoidaan solun keskipisteenä, vaikka se oikeasti sijaitsisi aivan sen nurkassa. Jos oletetaan, että ruudukon solut ovat kuutioita, saadaan enimmäisvirhe laskettua helposti Pythagoraan lauseella muotoon

$$\epsilon = \frac{a\sqrt{3}}{2}. \quad (4)$$

Vaatimus 6 esitti virheen enimmäissuuruudeksi yhtä millimetriä. Oktettipuu rajauslaatikon sivun kahtia joka tasolla ja ruudukko sen edelleen pieniin soluihin. Jos ole-

tetaan esimerkiksi pistepilven rajauslaatikko kuutioksi, jonka sivun pituus on sata metriä ja ruudukon sisältävän Scheiblauerin ja Schützin käyttämää 128 solua jokaiseen koordinaattiakselin suuntaan, saavutetaan puun tasolla 11 ruudukko, jonka solun sivun pituus on $a = \frac{100m/2^{10}}{128} \approx 0,763mm$. Tällaisessa ruudukossa enimmäisvirhe on $\frac{0,763mm \cdot \sqrt{3}}{2} \approx 0,661mm$, joten kaikki tähän ruudukkoon lisättyt pisteet täytyvät vaatimuksen 6. Tämä ei tarkoita sitä, että puu voisi sisältää pisteitä vain tasolta 11 alas päin. Pisteitä voi hyväksyä suuriinkin soluihin, jos ne ovat tarpeeksi lähellä sen keskipistettä.

Pisteiden sijainnin suhteellinen esitysmuoto nopeuttaa myös pistepilven käsittelyä. Laitossuunnittelussa pistepilveä joudutaan usein sovellukseen latauksen jälkeen liikuttamaan ja skaalaamaan, jotta se istuisi hyvin 3d-malliin. Jos pisteisiin olisi tallennettu absoluuttinen sijainti, jouduttaisiin pistepilveä tallennettaessa uudelleenkirjoittamaan kaikki pisteet käyttäjän määrittämällä transformaatiomatriisilla kerrottuna. Edellä kuvatulla esitysmuodolla transformaatio täytyy suorittaa vain oktettiipuun solmujen rajauslaatikoilla, joista tiivistetyn pisteen sijainti johdetaan. Tämä johtaa merkittävään parannukseen käyttökokemuksessa, sillä satojen miljoonien pisteiden kirjoittaminen levylle voi kestää kymmeniä minuutteja.

3.4 Tietorakenteen rakentaminen

Edellä todettiin sisäkkäispistepuiden sopivan pistepilven käsittelyyn käytettäväksi tietorakenteeksi laitossuunnitteluoohjelmistossa ja ehdotettiin pisteille kompaktimpaa esitysmuotoa. Käydään nyt läpi puun rakentamiseen käytettyjä algoritmeja.

Puun rakentaminen suoritetaan kahdessa vaiheessa. Ensin käydään kaikki syötepisteet läpi ja selvitetään niille rajauslaatikko. Tämän jälkeen luodaan tyhjä juurisolmu, jonka rajauslaatikkona toimii äsknen laskettu, kaikki pisteet sisältävä rajauslaatikko. Nyt syötepisteet voidaan lisätä yksitellen juurisolmuun, joka syöttää ne tarvittaessa edelleen lapsisolmuilleen. Rakentamisen ylintä tasoa on kuvattu algoritmissa 1.

Algoritmi 2 huolehtii pisteen pisteen lisäämisestä solmuun. Piste hyväksytään sol-

Algoritmi 1: RakennaOktettipuu

Syöte : Joukko pistepilviä P

Tuloste: Pisteet sisältävän oktettipuun juurisolmu s

```

1 // Selvitetään ensin pilvien rajauslaatikkojen unioni
2  $L \leftarrow$  tyhjä rajauslaatikko
3 for pistepilvi  $pc \in P$  do
4   for piste  $p \in pc$  do
5     if  $p \cap L = \emptyset$  then
6        $L \leftarrow L \cup p$ 
7     end
8   end
9 end

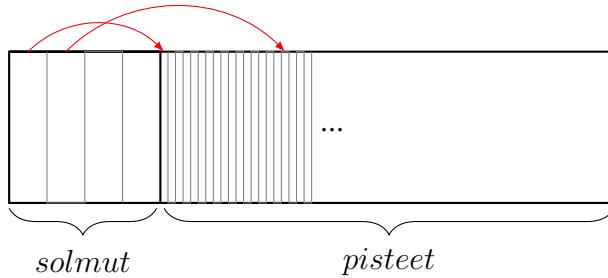
10 // Lisätään pisteet oktettipuuhun
11  $s \leftarrow$  juurisolmu, jonka rajauslaatikko on  $L$ 
12 for pistepilvi  $pc \in P$  do
13   for piste  $p \in pc$  do
14     LisääSolmuun( $s, p$ )
15   end
16 end

17 return  $s$ 

```

muun, jos sitä vastaava ruudukon solu on tyhjä, sen etäisyys solun keskipisteestä on pienempi kuin sallittu enimmäisvirhe ja solussa ei ole jo liikaa pisteitä. Solmujen sisältämien pisteiden määälle kannattaa asettaa yläraja, jotta saavutettaisiin sopiva haarautuminen. Luvussa 3.5 huomataan, että etenkin puun ylimmillä tasolla olisi renderöinnin kannalta hyvä, jos solmuissa olisi saman verran pisteitä.

Puun syvyyttä voi rajoittaa asettamalla ruudukon soluille vähimmäismitat. Yleensä lamerkeilaimen lähellä olevat pinnat on hyvin tiheästi näytteistettyjä ja tilannetta pahentaa,



Kuva 13: Tiedostoon tallennetaan ensin puun solmut, joiden jälkeen kaikki pisteet ovat peräkkäin taulukossa. Punaiset nuolet kuvavat solmuihin tallennettuja indeksejä pistetaulukkoon, josta siihen kuuluvat pisteet alkavat.

jos useat keilaimet ovat mitanneet samoja pintoja tiheästi. Algoritmin 2 rivillä 3 tarkastetaan, onko ylittäisikö uusi lapsisolmu puun enimmäissyyvyyden. Kun ennaltamäärätylle pohjatasolle hyväksytään kaikki pisteet, kunhan niitä vastaava ruudukon solu on vapaa, saadaan pistepilvelle tehokas ja globaali enimmäistiheys. Tällä tavalla pilveä saadaan harvnettua tehokkaasti ja vaatimus 3 voidaan tyydyttää.

Tietorakennetta tallennettaessa kirjoitetaan tiedostoon ensin tarvittavat tiedot puun solmuista ja pisteet vasta niiden jälkeen. Jokainen solmu sisältää tiedon siitä, kuinka monta pistettä siihen kuuluu, sekä ensimmäisen pisteen indeksin pistetaulukossa. Tämä mahdollistaa sen, että pistepilveä käsiteltäessä luetaan tiedostosta ensin vain kevyt hierarkia, jonka jälkeen vain tarvittavat pisteet voidaan lukea muistiin. Tiedoston rakennetta on havainnollistettu kuvassa 13. Pisteiden lukumäärän ja ensimmäisen pisteen indeksin lisäksi jokaisesta solmusta tallennetaan rajauslaatikko ja sijaintikoodi, joka kertoo sen sijainnin puussa. Sijaintikoodin numerot kertovat reitin juurisolmesta kyseiseen solmuun. Esimerkiksi koodi 014 tarkoittaa juurisolmun toisessa oktetissa sijaitsevan lapsen viidennenä oktetissa sijaitsevaa lasta. Lopuksi tarvitsee levylle kirjoittaa vielä millä puun tasolla se sijaitsee, jotta tiedostoa lukiessa tiedettäisiin sijaintikoodin pituus.

Yksittäinen solmu voidaan siis kirjoittaa levylle muodossa

```
struct Node {
    float min_x;
```

Algoritmi 2: LisääPisteSolmuun

Syöte : Puun solmu s ,

piste p , jolla on sijainti (x, y, z) ja väri (r, g, b)

```

1  $i \leftarrow$  sen ruudukon solun indeksi, jossa piste sijaitsee
2  $h \leftarrow$  hajautustaulu, johon solmun  $s$  pisteet tallennetaan
3 if Solmun  $s$  syvyys = puun enimmäissyvyys then
4   if  $i \notin h$  then
5      $h = h \cup (i, (r, g, b))$ 
6   else
7     Hylkää piste  $p$ 
8   end
9 else if  $s$  on täynnä then
10    $l \leftarrow$  uusi lapsisolmu
11   return LisääPisteSolmuun( $l, p$ )
12 else if  $i \in h$  then
13    $l \leftarrow$  uusi lapsisolmu
14   return LisääPisteSolmuun( $l, p$ )
15 else if  $\epsilon >$  ennalta määritetty enimmäisvirhe then
16    $l \leftarrow$  uusi lapsisolmu
17   return LisääPisteSolmuun( $l, p$ )
18 else
19   // Solmussa  $s$  ja sen ruudukon solussa  $i$  on tilaa, eikä virhe  $\epsilon$ 
      ole liian suuri. Piste voidaan lisätä solmuun  $s$ .
20    $h = h \cup (i, (r, g, b))$ 
21 end

```

```

    float min_y;
    float min_z;
    float max_x;
    float max_y;
    float max_z;
    unsigned char depth;
    unsigned char location_code[];
    unsigned int num_points;
    unsigned int point_index;
}

```

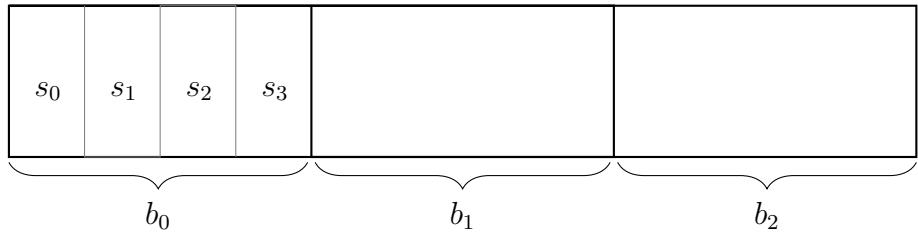
Nelitavuisilla liukuluvuilla ja kokonaisluvuilla vaatii solmu siis tallennustilaan $33+d$ tavua, missä d on solmun syvyys puussa.

3.5 Renderöinti

Nopein tapa renderöidä pistepilvi on pitää kaikki pisteen näytönohjaimen muistissa ja joka ruudunpäivityksellä piirtää kaikki näkymässä näkyvät pisteen. Vaikka koko pistepilvi mahtuisi näytönohjaimen muistiin, täytyy laitossuunnittelohjelmistossa visualisoida muitakin grafiikkaa. Näytönohjaimen muistia voidaan säädää käyttämällä niin kutsuttua puskurivirtaa (engl. *buffer streaming*). Yleinen ongelma grafiikan renderöinnissä on se, että näytönohjain renderöi dataa nopeammin kuin suoritin ehtii sille syöttää. Puskurivirran ajatuksena on pitää näytönohjain mahdollisimman toimeliaana jakamalla puskuri, jonka kautta dataa siirretään keskusmuistista näytönohjaimen muistiin, kolmeen osaan. Samalla kun näytönohjain käsittelee yhtä puskurin osaa, suoritin voi täyttää toista datalla, ja kolmas on jo palautumassa suorittimen täytettäväksi. Teoriassa näytönohjain voi vain vaihtaa luettavaa puskuria ilman, että sen tarvitsisi odottaa lankaan hidasta tiedonsiirtoa.

[33]

Kuvassa 14 pistepuskuri on jaettu osioihin b_0 , b_1 ja b_2 . Puskurin koko on valittu niin,



Kuva 14: Pistedatan renderöinnissä käytetty puskuri, joka on jaettu osioihin b_0 , b_1 ja b_2 . Osioon b_0 on kirjoitettu pisteet solmuista s_0-s_3 . Puskurivirran tarkoituksesta on vähentää tiedonsiirrosta aiheutuvaa latenssia.

että jokaiseen osioon mahtuu neljän täyden solmun pisteet. Käytännössä solmut eivät kuitenkaan aina ole täysiä, joten on mahdollista, että osiot jäävät vajaaksi. Puskureita kuitenkin vaihdetaan tiuhaan, joten ei ole järkevää käyttää aikaa puskurin täyttöasteen maksimointiin.

Tietorakenteen renderöintiä testatessa puskurivirta ei kuitenkaan tuottanut tarpeeksi miellyttävää visuaalista loppululosta. Etenkin pilveä pyöröiteltäessä ja sen läpi liikuttaessa puskurivirralla ehdittiin pilvestä renderöidä vain hyvin karkea tarkkuus niin, että ruudunpäivitystaajuus pysyi interaktiivisena. Ratkaisuna tähän näytti toimivan hyvin toinen pistepuskuri, jossa säilytetään karkeaa yleiskuvaa pilvestä. Tätä yleiskuvapuskuria päivitetään vain silloin, kun jotkut sen sisältämistä solmuista on pudonnut näkymän ulkopuolelle.

Algoritmissa 3 yleiskuvapuskuri täytetään ensimmäisellä renderöintikerralla niiden solmujen pisteillä, jotka ovat mahdollisimman ylhäällä puussa, ja ne ovat täynnä, eli sisältävät rakentamisvaiheessa määritellyn enimmäismäärän pisteitä. Seuraavilla renderöintikerroilla puskurista tarkistetaan, ovatko niiden sisältämät pisteet vielä näkyvissä. Jokaista pistettä ei tarvitse tarkastaa, jos ylläpidetään lista solmuista, joiden pisteet puskuriin lisättiin. Jos näkymä on muuttunut niin, että jokin solmu ei enää ole näkyvissä, vaihdetaan sen pisteet toisen solmun pisteisiin. Yleiskuvapuskurissa pidetään solmuja mahdollisimman läheltä puun juurta, jotta ne muodostaisivat tarpeeksi kattavan kuvan koko pistepilvestä.

Algoritmi 3: RenderöiYleiskuva

Syöte : Sisäkkäispistepuu P

```

1  $b_{yleiskuva} \leftarrow$  pistepuskuri, jota päivitetään vain tarvittaessa
2 if Ensimmäinen renderöintikerta then
3   // Täytetään yleiskuvapuskuri  $b_{yleiskuva}$ 
4   while  $b_{yleiskuva}$  ei ole täynnä do
5     // Käydään puuta läpi taso kerrallaan
6     for Solmu  $s \in P$  do
7       if  $s$  on täysi AND  $s$  on näkyvässä then
8         Lisää  $s$  puskuriin  $b_{yleiskuva}$ 
9       end
10      end
11    end
12 else
13   // Päivitetään yleiskuvapuskuri  $b_{yleiskuva}$ 
14   for Solmua  $s$  vastaavat pisteet puskurissa  $b_{yleiskuva}$  do
15     if Solmu  $s$  ei ole enää näkyvässä then
16       Vaihda pisteet toisiin, mahdollisimman ylhäällä puussa olevan solmun
17       pisteisiin
18     end
19   end
20 Renderöi puskuri  $b_{yleiskuva}$ 

```

Yleiskuvan renderöinnin jälkeen pistepilveä tarkennetaan renderöimällä loput pisteet puskurivirralla. Algoritmi 4 käy puun solmuja läpi taso kerrallaan ja lisää solmun pisteet täyttövuorossa olevaan puskurivirran osioon, jos solmua ei ole jo renderöity yleiskuva-puskurilla. Puskuriosion täyttyessä vaihdetaan täytettäväksi seuraava osio. Puskurivirras-

Algoritmi 4: RenderöiPuskurivirta

Syöte : Sisäkkäispistepuu P

Yleiskuvapuskuri $b_{yleiskuva}$

```

1  $B \leftarrow$  pistepuskuri, joka on jaettu osioihin  $b_0, b_1, b_2$ 
2  $i \leftarrow 0 //$  Täytettävän puskuriosion indeksi
3  $S \leftarrow$  näkyvissä olevat puun solmut
4 Järjestää  $S$  ruudulle projisoidun koon mukaan
5 for Solmu  $s \in S$  do
6   if Puskuriosiossa  $i$  ei ole tilaa täydelle solmulle then
7     Lähetä näytönohjaimelle puskuriosio  $i$ 
8     Lähetä näytönohjaimelle osion  $i$  synkronointiobjekti
9      $i = (i + 1) \text{ mod } 3$ 
10    Odota, että näytönohjain on käsitellyt osion  $i$  synkronointiobjektin
11  else if Solmun  $s$  pisteet eivät ole puskurissa  $b_{yleiskuva}$  then
12    Lisää solmun  $s$  pisteet puskuriosioon  $i$ 
13 end

```

sa täytyy varmistaa, ettei sama osio ole sekä suorittimen kirjoitettavana, että näytönohjaimen luettavana. Tämä tapahtuu lähetämällä näytönohjaimelle jokaisen puskuriosion täytymisen jälkeen synkronointiobjekti (engl. *sync object*). Kun uutta osiota otetaan kirjoittavaksi, tarkastetaan, että näytönohjain on merkannut kyseisen osion synkronointiobjektin käsitellyksi. [34]

Algoritmi 4 suorittaa pistepilvelle näkyvyyskarsintaa (engl. *visibility culling*). Puu käydään ensin kokonaan läpi ja renderöitäviksi valitaan vain ne solmut jotka ovat täyssin tai osittain näkymäkartion (engl. *view frustum*) sisällä. Näkyvyyskarsinnan lisäksi algoritmin voidaan katsoa suorittavan yksinkertaista yksityiskohtien karsintaa (engl. *detail culling*), kun kuvaruudulle suurena projisoidut solmut renderoidään ensin.

Yksityiskohtien karsinta on suosittu nopeutustekniikka tietokonegrafiikassa. Ajatuksesta on renderöidä vain tärkeimmät objektit ja jättää kaukana olevat tai pienet objektit renderöimättä, jotta ruudunpäivitystaajuus pysyy interaktiivisena. Yksinkertainen tapa karsia yksityiskohtia on järjestää objektit niiden kuvaruudulle projisoidun koon mukaan ja renderöidä niitä tiettyyn rajaan asti, tai kunnes aika loppuu kesken. Akenine-Möller et al. [35] esittävät kuvaruudulle projisoidun objektiin koon arviolle kaavaa

$$a = \pi \left(\frac{nr}{\mathbf{d} \cdot (\mathbf{c} - \mathbf{v})} \right)^2, \quad (5)$$

jossa n on katselupisteen etäisyys kuvatasosta, r objektiin rajauspallon säde ja \mathbf{c} keskipiste, \mathbf{d} on normalisoitu katsomissuunta, ja \mathbf{v} katselupiste. [23]

Mikko Yllikäinen huomautti pro gradu -tutkielmassaan [23], ettei tarkkaa arvioita objektiin koosta kannata selvittää, jos halutaan selville vain suuruusjärjestys. Yllikäinen yksinkertaistaa kaavan muotoon

$$\begin{aligned} a &= \frac{r}{\mathbf{d} \cdot (\mathbf{c} - \mathbf{v})} \\ &\approx \frac{r}{\sqrt{(c_x - v_x)^2 + (c_y - v_y)^2 + (c_z - v_z)^2}} \\ &\propto \frac{r}{(c_x - v_x)^2 + (c_y - v_y)^2 + (c_z - v_z)^2}. \end{aligned} \quad (6)$$

Yllikäinen nimittää tällä kaavalla muodostetu suuruusjärjestyksen käyttämistä yksityiskohtien karsinnassa kontribuutiokarsinnaksi (engl. *contribution culling*). On huomattava, että tällainen karsinta ei ole mahdollista paralleliprojektiomaisemissa, joissa katseluetäisyys ei vaikuta objektien kokoon. [23]

3.6 Pisteiden valinta

4 Tietorakenteen arviointi

4.1 Tietorakenteen rakentaminen ja lataaminen

4.2 Visualisointi

4.3 Pisteiden valitseminen

5 Johtopäätökset

Viitteet

- [1] T. W. Marc Levoy, Technical report, Computer Science Department, University of North Carolina at Chapel Hill (unpublished).
- [2] T. Qu ja W. Sun, Journal of Civil Engineering and Architecture **9**, 1269 (2015).
- [3] H. Rauno *et al.*, Technical report, Tiehallinto (unpublished).
- [4] C. Scheiblauer ja M. Pregebaumer, in *Proceedings of the 16th International Conference on Cultural Heritage and New Technologies* (PUBLISHER, 2011), pp. 242–247.
- [5] v. . Verkkolähde, <https://www.carnuntum.at/en/science-history/carnuntum-in-roman-times>.
- [6] F. Menna *et al.*, ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **XLI-B5**, 675 (2016).
- [7] J. Hecht, Optics and Photonics News **29**, 26 (2018).
- [8] J. Huhtanen, Helsingin sanomat .
- [9] L. G. AG, 2018.
- [10] J. Fabritius, Opinnäytettyö, Tampereen ammattikorkeakoulu, 2009.
- [11] H. Houshiaar, J. Elseberg, D. Borrmann ja A. Nuchter, Geo-spatial Information Science **18**, (2015).
- [12] v.-h. v. . 3DTK, The 3D Toolkit.
- [13] P. J. Besl ja N. D. McKay, IEEE Transactions on Pattern Analysis and Machine Intelligence **14**, 239 (1992).
- [14] C. Harris ja M. Stephens, in *In Proc. of Fourth Alvey Vision Conference* (PUBLISHER, 1988), pp. 147–151.
- [15] E. Rosten ja T. Drummond, in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* (PUBLISHER, 2005), Vol. 2, pp. 1508–1515 Vol. 2.
- [16] D. G. Lowe, in *Proceedings of the Seventh IEEE International Conference on Computer Vision* (PUBLISHER, 1999), Vol. 2, pp. 1150–1157 vol.2.
- [17] M. Weinmann, *Reconstruction and Analysis of 3D Scenes: From Irregularly Distributed 3D Points to Object Classes*, 1st ed. (Springer Publishing Company, Incorporated, 2016).
- [18] C. M. Huang ja Y.-H. Tseng, in *29th Asian Conference on Remote Sensing 2008, ACRS 2008, 29th Asian Conference on Remote Sensing 2008, ACRS 2008* (PUBLISHER, 2008), pp. 1925–1930.

- [19] M. Berger *et al.*, Computer Graphics Forum (2016).
- [20] J. Giesen ja F. Cazals, (2006).
- [21] M. Piipponen, Opinnäytettyö, Satakunnan ammattikorkeakoulu, 2012.
- [22] Aveva, AVEVA Laser Modeller.
- [23] M. Yllikäinen, Master's thesis, 2013.
- [24] CADMATIC, Keskustelut CADMATIC:n henkilöstön kanssa, 2019.
- [25] S. Rusinkiewicz ja M. Levoy, Proceedings of SIGGRAPH **2000**, (2001).
- [26] C. Dachsbacher, C. Vogelsgang ja M. Stamminger, ACM Transactions on Graphics **22**, 657 (2003).
- [27] M. Wimmer ja C. Scheiblauer, in *Proceedings Symposium on Point-Based Graphics 2006*, Eurographics (Eurographics Association, 2006), pp. 129–136.
- [28] J. Yang, R. Li, Y. Xiao ja Z.-G. Cao, in *3D reconstruction from non-uniform point clouds via local hierarchical clustering* (PUBLISHER, 2017), p. 1042038.
- [29] J. Davidsson, Technical report, 3point Oy, Lapinlahdenkatu 16 00180 Helsinki (unpublished).
- [30] C. Scheiblauer, Ph.D. thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2014.
- [31] M. Sch Master's thesis, .
- [32] v. . Verkkolähde, <https://www.nvidia.com/en-gb/design-visualization/quadro/rtx-8000/>.
- [33] v. . Verkkolähde, https://www.khronos.org/opengl/wiki/Buffer_Object_Streaming.
- [34] v. . Verkkolähde, https://www.khronos.org/opengl/wiki/Sync_Object.
- [35] T. Akenine-Moller, T. Moller ja E. Haines, *Real-Time Rendering*, 2nd ed. (A. K. Peters, Ltd.Natick, MA, USA, 2002).