Pistepilvien visualisointi laitossuunnitteluohjelmistossa

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

TURUN YLIOPISTO
Tulevaisuuden teknologioiden laitos
Timo Heinonen Pistepilvien visualisointi laitossuunnitteluohjelmistossa
Pro Gradu, 16 s., 3 liites.
Tietojenkäsittelytiede
10. marraskuuta 2019
Turun yliopiston laatujärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check -järjestelmällä.

Juuh elikkäs gradutyötä...

Sisältö

1 Johdanto **1**

1.1 Pistepilvet 1

1.2 Laserkeilaimet 2

1.3 Pistepilvidatan käsittely 4

1.4 Pistepilvien hyödyntäminen laitossuunnittelussa 5

2 Pistepilvien käsittelyssä käytetyt tietorakenteet **6**

2.1 Vaatimukset ja haasteet 7

2.2 Hierarkiset tietorakenteet 8

2.2.1 QSplat 8

2.2.2 Peräkkäispistepuut 10

2.2.3 Sisäkkäispistepuut 11

3 Laitossuunnittelusovellukseen optimoitu tietorakenne **15**

4 Vertailu **15**

5 Johtopäätökset **15**

1 Johdanto

Tässä luvussa määritellään pistepilven käsite ja esitetään sille sovelluskohteita.

1.1 Pistepilvet

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

Pistepilville on useita käyttökohteita. Historiallisten rakennusten ja muistomerkkien säilyminen jälkipolville voidaan varmistaa luomalla niistä 3d-malli. [2] Rakennuksen manuaalinen mallintaminen olisi hyvin suuri työ verrattuna muutaman kymmenen pistepilven luomiseen laserkeilaimella, mikä voidaan tehdä päivässä. Toinen yleinen pistepilvien sovelluskohde on arkkitehtuuri ja rakentaminen. [3] Jos rakennelmaan halutaan tehdä muutostöitä, on siitä yleensä tehtävä 3d-malli. Jälleen laserkeilaus on mallintamista huomattavasti edullisempi vaihtoehto.

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 tienkäyttäjiin ja esteisiin. Itseohjautuvat autot ovat merkittävä tutkimuskohde myös pistepilvien käsittelyn kannalta. Auton katolle asennettavan laserkeilaimen tulisi olla tarkka, nopea ja edullinen, ja sen tuottamaa pistedataa täytyy voida käsitellä reaaliajassa. [8]

[1] 1980-luvulla pisteitä ehdotettiin jopa yleisiksi renderöintiprimitiiveiksi kuvaamaan mitä tahansa geometriaa [13]. Aika kuitenkin näytti kolmion olevan tehokkaampi primitiivi ja nykyään grafiikkakirjastot ja -prosessorit onkin optimoitu kolmioiden käsittelyyn.

[2] ks. esim. [12]

[3] ks. esim [15]

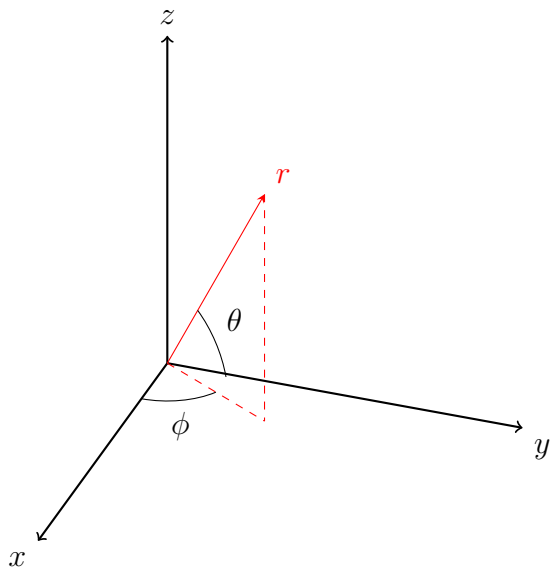
Kuva 1. Kulkuaikatekniikkaan perustuva Leica RTC360 -laserkeilain. Kuva: https://leica-geosystems.com/-/media/images/leicageosystems/about-us/news%20room/reporter/reporter-83/09-discovering-the-power-of-scanning/leica-espresso_expert_insights_640x750_slider4.ashx?la=en&hash=9D35592CBB571C3777E1E8A9A0A056BD

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 tarkemmilla laserkeilamilla myös rakennuksista. [11]

1.2 Laserkeilaimet

Perinteinen laserkeilain on jalustalla seisova laite, joka pyörii pystyakselinsa ympäri ampuen laserpurskeita ympärilleen tihein välein. Laserkeilain mittaa etäisyyksiä pisteisiin, joista laserpurske heijastuu takaisin keilaimeen ja muodostaa näistä pisteistä pistepilven. Heijastuksen voimakkuutta käytetään usein määräämään pisteelle väri.

Yleensä tarkasteltavaa kohdetta täytyy keilata useista eri suunnista, jotta saataisiin tarpeeksi kattava joukko pistepilviä. Kohteesta riippuen voidaan tarvita jopa satoja keilauk-



Kuva 2. Pallokoordinaatisto

sia. Pistepilvien sovittamista yhteen tarkastellaan luvussa 1.3.

Nykyaikaisella laserkeilaimella saadaan muodostettua hyvin tarkka ja tiheä pistepilvi nopeasti. Esimerkiksi kuvassa 1.1 näkyvä Leica Geosystemsin RTC360 -keilaimen luvataan mittaamaavan jopa kaksi miljoonaa pistettä sekunnissa ja kiertävän täyden ympyrän alle kahdessa minuutissa. Keilaimen lisäksi laitteessa on kamera, jolla saadaan määritettyä pisteille oikeat värit. [1]

Laserkeilaimien toimintaperiaatteissa on eroja. Kaksi yleisintä toimintaperiaatetta ovat kulkuaikatekniikka *(engl. Time-Of-Flight)* ja vaihesiirtotekniikka *(engl. phase shift)*. Kulkuaikatekniikkassa pisteen etäisyys keilaimesta selviää ajasta, joka kuluu laserpurskeen lähetettämisestä sen heijastuksen vastaanottamiseen. Pisteen etäisyys keilaimesta on yksinkertaisesti $d = \frac{c \cdot t}{2}$, missä c on valonnopeus ja t on mitattu aika. [6] Vaihesiirtotekniikka perustuu keilaimesta lähtevän signaalin vaiheen vertaamista palaavan signaalin vaiheeseen. Pisteen etäisyys keilaimesta saadaan laskemalla $d = n \cdot \lambda + \frac{\Phi \cdot \lambda}{2 \cdot \pi}$, missä n on havainnon täysien aaltojen määrä, λ on signaalin aallonpituus ja Φ on lähtevän ja palaavan signaalin vaihe-ero. [6]

Laserkeilain tallentaa mittaamansa pisteen pallokoordinaateissa. Pisteen siirtäminen pallokoordinaateista karteesiseen koordinaatistoon onnistuu laskemalla koordinaatit $x =$

$r \cdot \sin\theta \cdot \cos\phi, y = r \cdot \sin\theta \cdot \cos\phi, z = r \cdot \cos\theta$, missä r on pallon säde, θ on korotuskulma ja ϕ atsimuuttikulma. Pallokoordinaatistoa on havainnollistettu kuvassa 2.

1.3 Pistepilvidatan käsittely

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 pistepilvien rekisteröinniksi.

Pistepilvet voidaan reksiteröidä usealla eri tavalla. Joskus keilattavaan kohteeseen asetetaan erityisiä merkkikuvioita, jotka näkyvät useasta keilaimesta. Kun tiedetään merkkien etäisyys ja suunta kustakin keilaimesta, voidaan pistepilvet sovittaa yhteen koordinaatistoon. Joissakin sovelluksissa käyttäjä merkitsee pilvistä pisteitä, jotka kuvaavat samaa aluetta.

Pistepilviä voidaan rekisteröidä myös ilman merkkikuvioita. Iteratiivinen lähimmän pisteen algoritmi *(engl. iterative closest point, ICP)* sovittaa pistepilven toiseen etsimällä rotaation ja translaation, jolla pilvien välinen virhe saadaan minimoitua. ICP-algoritmi määrittää ensin pilvistä toisiaan vastaavat pisteet ja virhe lasketaan kaikkien vastaavuuksien välisistä etäisyyksistä. Yksinkertaisimmillaan pistettä vastaavaksi pisteeksi merkitään sovitettavan pilven lähinnä sijaitseva piste. ICP-algoritmi tarvitsee käyttäjältä usein hyvän alkuarvauksen, jotta pilvien sovittaminen onnistuisi. [4]

Pistepilvissä on usein muukalaispisteitä *(engl. outlier)*, johtuen esimerkiksi laserkeilaimen epätarkkuudesta tai vaikkapa tuulen heiluttamista puiden lehdistä. Yksinkertainen tekniikka poistaa muukalaispisteitä pilvestä on verrata pisteen normaalivektoria sen naapuripisteiden normaaleihin. Pisteen p_i normaalivektori voidaan selvittää pääkomponenttianalyysillä *(engl. principal component analysis, PCA)*. Ensin on etsittävä pilvestä pisteen p_i k lähintä naapuria, jonka jälkeen naapurston pisteistä lasketaan ominaisarvot.

[4] Täysin automaattista pistepilvien rekisteröintiä on tutkittu paljon, ks. esim. [19].

Kahta suurinta ominaisarvoa vastaavaa ominaisvektoria voidaan käyttää kuvaamaan tasoa, joka sovitetaan naapuruston päälle. Jäljelle jäävä ominaisvektori kuvaa pisteen p_i normaalia. [10]

Joissakin sovelluksissa halutaan luoda pistedatasta kohteen pintoja kuvaava polygoniverkko, jotta visualisointi olisi nopeampaa olemassaolevilla grafiikkakirjastoilla ja visuaalinen lopputulos parempi. Yksinkertainen tekniikka luoda tiivis kolmiointi pistepilvestä on Delaunayn kolmiointi[5] *(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 pisteiden välillä on kaari, mikäli niitä vastaavat Voronoin solut jakavat Voronoin janan. [7]

Joskus pistepilvet halutaan kompaktimpaan esitysmuotoon, jossa niiden käsittely on helpompaa. Pistepilvestä voidaan luoda panoramakuva projisoimalla laserkeilaimen ympäröimät pisteet kaksiulotteiselle kuvatasolle.[6] Panoramakuvan resoluutiosta riippuen pistepilvestä voidaan karsia huomattava määrä pisteitä, joiden koko olisi liian pieni projisoituna kuvatasolle. Panoramakuvaan on helppo soveltaa erilaisia kuvankäsittelyalgoritmeja, kuten piirteentunnistusta *(engl. feature detection)*.

1.4 Pistepilvien hyödyntäminen laitossuunnittelussa

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

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

[6]Projektio vaikuttaa suuresti panoramakuvan laatuun. ks. esim [9]

plant design software).

Tietokoneavusteisessa suunnittelussa pistepilviä käytetään olemassaolevien rakenteiden taltiointiin. Usein käytetty esimerkki on autoteollisuuden alalta: ryhmä suunnittelijoita kokeilee uutta korimallia rakentamalla prototyyppiauton helposti muovattavasta materiaalista. Kun prototyyppi on todettu aerodynaamiseksi ja miellyttävän näköiseksi, täytyy se saada digitoitua, jotta se voidaan siirtää massatuotantoon. Usein yksinkertaisin ja kustannustehokkain tapa on laserkeilata prototyyppi ja jatkokäsitellä 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 suunnitteluohjelmalla. [14]

Pistepilviä voi hyödyntää monin tavoin laitossuunnittelussa. Vähiten töitä vaatii pilvien käyttö sellaisenaan. Jos laitokseen halutaan vaikkapa uusi vesiputki, voidaan sen mahtuminen varmistaa reitittämällä putki suunnitteluohjelmistolla ja tarkastamalla, osuuko se pistepilveen. Jos vanhasta laitoksesta halutaan luoda ajantasalla oleva 3d-malli, voi suunnittelija mallintaa laitosta pistepilvien avulla. Pistepilven päälle on helppo sijoittaa suunnitteluohjelman putkistoja ja laitteita oikeille paikoilleen. Markkinoilla on myös ohjelmistoja, joiden luvataan tuottavan pistepilvestä automaattisesti älykäs 3d-malli komponenttitietoineen ilman aikaavievää päällemallinnusta [2].

2 Pistepilvien käsittelyssä käytetyt tietorakenteet

Tässä luvussa selvitetään, mitä vaatimuksia pistepilvien käsittely asettaa tietorakenteille ja visualisointialgoritmeille, tutustutaan aiheesta tehtyihin julkaisuihin ja esitetään laitossuunnitteluohjelmistolle optimoitu tietorakenne.

2.1 Vaatimukset ja haasteet

Suurin haaste pistepilvien käsittelyssä on niiden koko. Nykyaikainen laserkeilain, kuten luvussa 1.2 esitetty Leica Geosystemsin RTC360, tuottaa pistepilven, jossa on satoja miljoonia pisteitä. Kun tällaisella keilaimella tehdään useita keilauksia, on pisteiden määrä valtava. Oletetaan esimerkiksi, että suuressa 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, joka saattaisi vielä mahtua tehokkaan työaseman keskusmuistiin, mutta ei grafiikkaprosessorin muistiin. Yleensä koko 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 visualisoida reaaliajassa. Pisteiden määrää voidaan karsia harventamalla pilveä esimerkiksi jakamalla 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 reaaliajassa. Interaktiivisessa ohjelmistossa, kuten laitossuunnitteluohjelmistossa, on tärkeää kuitenkin pitää ruudunpäivitystaajuus tarpeeksi korkeana. Tällöin voidaan pistepilvestä piirtää ruudulle ensin karkea yleiskuva, jota tarkennetaan vähitellen, ellei käyttäjä keskeytä visualisointia esimerkiksi vaihtamalla kuvakulmaa. Tällainen astettainen visualisointi on mahdollista käyttämällä hierarkisia tietorakenteita pistepilven käsittelyssä. Tämän tekniikan etuna on se, että käyttäjä näkee välittömästi pistepilven yleisen muodon ilman, että hänen täytyy odottaa, että koko pilvi on visualisoitu. Jos käyttäjällä riittää kärsivällisyys, näkee hän kaikki pilven yksityiskohdat, kun visualisointialgoritmi on käynyt koko tietorakenteen lä-

pi.

Tämän tutkielman osana kehitetään laitossuunnitteluohjelmistolle optimoitu hierarkinen tietorakenne pistepilvien käsittelyyn. Esitetään tietorakenteelle seuraavat vaatimukset:

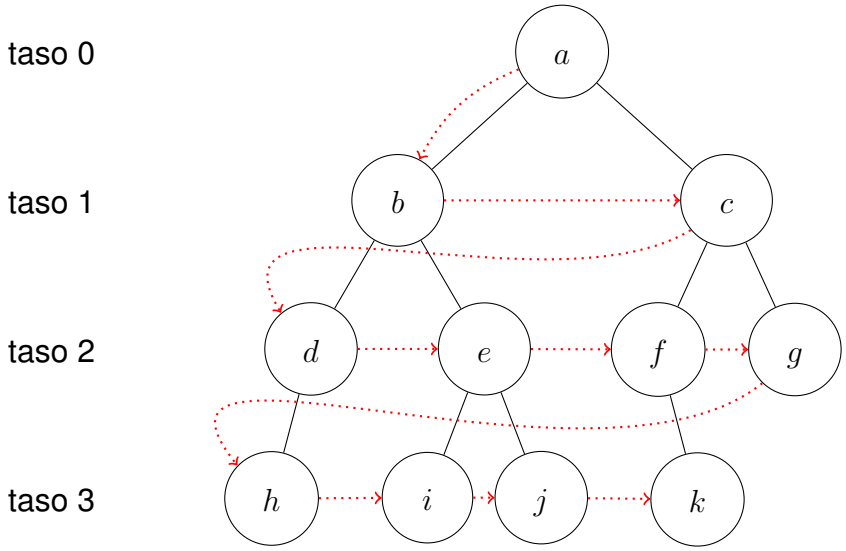
- On voitava visualisoida karkea yleiskuva pistepilvestä vain pienellä osalla datasta.
- Pistepilveä on voitava harventaa siten, että tarkkuuden menetystä voidaan kontrolloida.
- On käytettävä ulkoisen muistin algoritmeja, eli koko pilveä ei pidetä kerralla keskusmuistissa.
- Hakuoperaatioiden tulee onnistua logaritmisessa ajassa pisteiden määrään suhteutettuna.
- Käyttäjän on voitava määrittää pilvestä alueita, joiden sisältävien pisteiden ominaisuuksia, kuten näkyvyyttä tai väriä, voidaan muuttaa.

2.2 Hierarkiset tietorakenteet

2.2.1 QSplat

Yksi ensimmäisistä pistedatan visualisointiin käytetyistä hierarkisista tietorakenteista on Rusinkiewiczin ja Levoyn esittämä QSplat, joka on kehitetty polygoniverkon visualisointiin pisteiden avulla. Tietorakenne muodostetaan kolmioidusta mallista, jossa kolmioiden normaalit tunnetaan, joten se ei suoraan sovellu raa'an pistepilvidatan käsittelyyn.[7] QSplatissa on käytetty kuitenkin monia kiinnostavia tekniikoita, joita voi hyödyntää pistepilvien käsittelyssä. [16]

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



Kuva 3. Puun läpikäyntijärjestys (punainen katkoviiva) muodostaa luonnolliset tarkkuustasot

QSplat perustuu puurakenteeseen, jonka solmuissa on avaruutta rajaavia palloja *(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 kolmio. Siitä muodostetaan lehtisolmu, jonka rajaava pallo 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. [16]

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 muistin tekniikoihin, voidaan täplien piirtäminen aloittaa heti, kun tarpeeksi puun solmuja on ladattu levyltä muistiin.[8] Puun rakennetta on havainnollistettu kuvassa 3. [16]

Toinen hyödyllinen QSplatissa käytetty tekniikka on koordinaattien kvantisointi *(engl.*

[8]Rusinkiewicz ja Levoy päättävät ruudulle projisoitujen täplien koon perusteella kuinka syvälle puussa tulee edetä.

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. Pallon säde 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 hakutaulua, voidaan pallon sijainti esittää vain 13:lla bitillä, kun normaali liukulukuesitys vaatisi vähintään 16 tavua. [16]

QSplat onnistui visualisoimaan 1,5-2,5 miljoonaa pistettä sekunnissa, mikä on sen aikaisella laitteistolla erinomainen tulos [16]. Kuten sanottu, se ei sellaisenaan kuitenkaan sovellu laserkeilattujen pistepilvien käsittelyyn. Pistepilvien pisteiden normaaleja ei yleensä tiedetä, joten ne pitäisi esiprosessointivaiheessa selvittää esimerkiksi luvussa 1.3 esitetyllä tekniikalla. QSplat tarjoaa kuitenkin monia tekniikoita, joita pistepilviä käsittelevässä 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 rajaavat pallot. Täplien väri määräytyy lapsisolmujen värien keskiarvolla. [4]

Jokaiseen puun solmuun on virhelaskelmien perusteella lisätty rajat katseluetäisyydelle, jolla solmu valitaan visualisoitavaksi. Visualisointivaiheessa hierarkia litistetään taulukoksi, joka voidaan syöttää suoraan grafiikkaprosessorille. Grafiikkaprosessori käy taulukkoa läpi ja valikoi sopivat solmut, joiden perusteella täpliä piirretään ruudulle. [4]

Peräkkäispistepuut voidaan visualisoida nopeasti grafiikkaprosessorin käytön ansios-

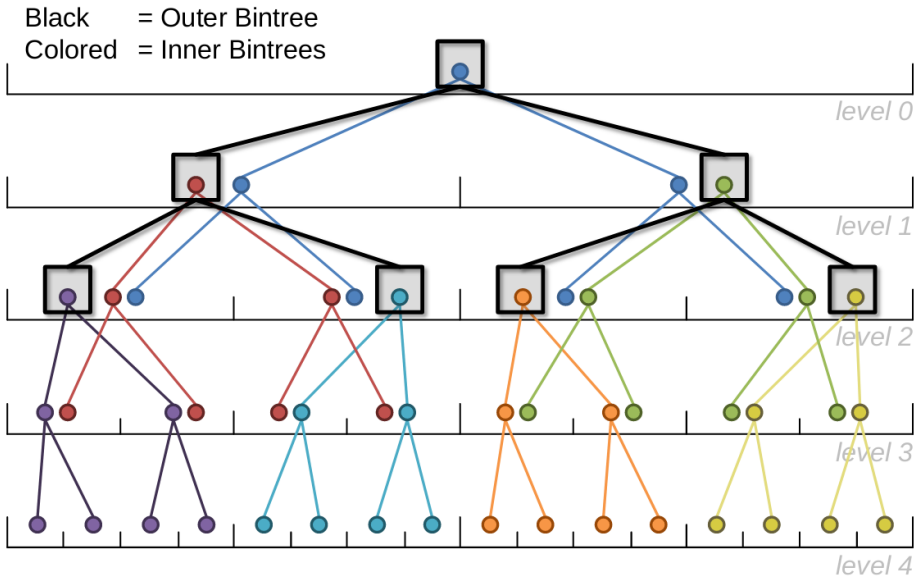
ta, mutta niissä on myös heikkouksia. Tietorakenteen vaatimuksena on, että kaikki data mahtuu grafiikkaprosessorin muistiin. Näin on vain pienillä malleilla ja tilannetta pahentaa se, että peräkkäispistepuut eivät ole kovin säästäväisiä muistin suhteen. Hierarkian jokaisessa sisäsolmussa luodaan lisää dataa, kun lapsisolmujen unionia kuvataan täplän sijainnilla ja koolla, sekä keskimääräisellä värillä.

Wimmer ja Scheiblauer esittävät parannuksia peräkkäispistepuihin. Uusien täplien luomisen sijaan puun sisäsolmuissa valitaan lapsisolmuista edustaja, joka parhaiten kuvaa sisäsolmun esittämää avaruuden osaa. Tämä on tärkeä huomio, sillä käsiteltäessä massiivisia pistepilviä tulisi välttää ylimääräisen datan luomista. Hierarkiasta muodostetaan tarkkuustasot siten, että alemmat tarkkuustasot sisältyvät ylempiin tasoihin ja visualisoitaessa oikea tarkkuustaso valitaan täplien koon ja katseluetäisyyden perusteella. Solmua vastaavan täplän koko määräytyy siitä, kuinka syvällä hierarkiassa se on. Wimmer ja Scheiblauer kutsuvat tätä rakennetta muistioptimoiduksi peräkkäispistepuuksi. *(engl. memory optimized sequential point tree, MOSPT)* [20]

2.2.3 Sisäkkäispistepuut

Wimmer ja Scheiblauer kritisoivat muistioptimoituja peräkkäispistepuita siitä, että ne eivät tue näkökentän ulkopuolisten pisteiden tehokasta karsimista ja siitä, ettei muistioptimointi yksinään riittänyt poistamaan tarvetta ulkoisen muistin algoritmeille. Ratkaisuksi he esittivät sisäkkäisiä oktettipuita *(engl. nested octree)*. Oktettipuu[9] on yksinkertainen avaruutta rekursiivisesti jakava tietorakenne, jonka jokainen sisäsolmu jakaa kuvaamansa avaruuden osan kahdeksaan osaan. Wimmerin ja Scheiblauerin tietorakenteessa oktettipuita on kahdessa tasossa. Ulompaa oktettipuuta käytetään avaruuden jakamiseen, sen tehokkaseen 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 sol-

[9]Tätä suomennosta on käyttää esimerkiksi [5]



Kuva 4. Sisäkkäinen binääripuu, jossa sekä ulomman, että sisempien puiden syvyys on kolme. Ulompaa puuta kuvaavat mustat neliöt ja sisempää värikkäät ympyrät. Puista muodostuu viisi tarkkuustasoa. Kuva: [18]

muun. [20]

Sisäkkäisistä oktettipuista luodaan tarkkuustasot siten, että sisemmistä puista kerätään pisteitä ulomman puun tasojen mukaan. Tarkkuustasoon kuuluvat pisteet sijaitsevat siis ulomman puun samalla tasolla, mutta useiden sisempien puiden eri tasoilla. Tarkkuustasojen muodostumista on havainnollistettu kuvassa 2.2.3. Puut tallennetaan levylle tarkkuustaso kerrallaan, mikä mahdollistaa ulkoisen muistin algoritmien käytön. Visualisoitaessa tarvitsee levyltä lukea pisteitä vain haluttuun tarkkuustasoon asti, eikä loppuja pisteitä tarvitse ladata muistiin. [20]

Scheiblauer jalostaa sisäkkäisten oktettipuiden ideaa väitöskirjassaan esittelemällä muokattavat sisäkkäiset oktettipuut *(engl. modifiable nested octree, MNO)*. Jos edellä esiteltyjä 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. [18]

MNO:n rakenne eroaa sisäkkäisistä oktettipuista siten, että sisemmät puut korvataan säännöllisillä, kolmiulotteisilla ruudukoilla, joihin pisteet tallennetaan. MNO:n rakenta-

minen 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 tarpeeksi, jotta olisi järkevää luoda uusia solmuja puuhun. Kun ennaltamäärätty vähimmäismäärä pisteitä on kertynyt ylimääräisten pisteiden taulukkoon, luodaan ruudukon sisältävälle solulle lapsisolmuja ja sijoitetaan ylimääräiset pisteet niihin. Ruudukkoon sijoitettavien pisteiden määrälle on hyvä asettaa myös yläraja. [18]

Jokainen tietorakenteen solmu tallennetaan omaan tiedostoonsa levylle, josta niitä ladataan muistiin visualisointivaiheessa tarvittaessa. Visualisointialgoritmiin kuuluu käyttäjän asettama pistebudjetti, joka asettaa ylärajan yhdessä ruudunpäivityksessä piirrettävien pisteiden määrälle.[10] Tätä rajaa säätämällä käyttäjä saa jonkinlaisen kontrollin ruudunpäivitystaajuuden suhteen. [18]

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 tapauksessa juurisolmulle luodaan vanhempia kunnes jokin niistä muodostaa tarvittavan kokoisen avaruuden, ja piste lisätään sen ruudukkoon. Kun puun vanhan juuren yläpuolelle luodaan uusia solmuja, jää niiden ruudukot vajaaksi. Tällöin alemmista solmuista nostetaan pisteitä ylöspäin niin kauan, kunnes vajaita ruudukoita on vain lehtisolmuissa. Pisteiden poistaminen puusta on triviaalia, kun sisäsolmuihin mahdollisesti jäävät tyhjät ruudukot täytetään kuten pisteitä lisättäessä. [18]

Markus Schütz jatkoi Wimmerin ja Scheiblauerin työtä esittelemällä opinnäytetyössään verkkoselaimessa ajettavan Potree-nimisen pistepilvivisualisoijan. Potreen käyttämä tietorakenne perustuu Scheiblauerin muokattaviin sisäkkäisiin oktettipuihin, mutta hierarkian rakennusvaiheessa kiinnitetään huomiota pisteiden tasaiseen jakautumiseen solmu-

[10] Scheiblauer testasi pistepilvivisualisoijaansa asettamalla rajan vain sataantuhanteen pisteeseen.

jen välille. Oktettipuun sisäsolmujen ruudukoihin hyväksytään uusia pisteitä vain, jos ne ovat tarpeeksi kaukana muista ruudukon pisteistä. Lehtisolmut hyväksyvät ennaltamäärättyyn rajaan saakka kaikki pisteet, kunnes ne muutetaan sisäsolmuiksi ja liian lähekkäin olevat pisteet jaetaan uusien lapsisolmujen kesken. [17]

Potree käyttää ulkoista muistia tehokkaasti ja pystyy käsittelemään jopa 640 miljardia pistettä sisältäviä pistepilviä.[11] 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. Näin voidaan välttää sekä turhien pisteiden, että näkökentän ulkopuolella olevien hierarkian haarojen lataaminen muistiin. [17]

Potreen visualisointialgoritmi priorisoi niitä hierarkian solmuja, jotka ovat lähellä katselupistettä ja joiden kuvaruudulle projisoitu koko on suurin. Visualisoinnin suorituskykyä voidaan säädellä Scheiblauerin toteutuksen mukaisesti käyttäjän asettamalla pistebudjetilla. Schütz on kehittänyt Potreehen myös hienostuneen, grafiikkaprosessorilla ajettavan algoritmin mukautuvaan pisteiden koon määrittämiseen; pistepilven harvemmissa osissa piirretään pisteet suurempina, jottei reikiä esiintyisi. [17]

[11] Kyseinen pistepilvi (Actueel Hoogtebestand Nederland, ANH2, `http://ahn2.pointclouds.nl/`) kuvaa koko Alankomaiden valtiota ja se vaatii 7,68 teratavua tallennustilaa. Potreen tietorakenteessa pistepilvi jakautui 13:lle tasolle ja 38:aan miljoonaan solmuun.

3 Laitossuunnittelusovellukseen optimoitu tietorakenne

4 Vertailu

5 Johtopäätökset

Viitteet

[1] Leica Geosystems AG, 2018.

[2] Aveva. Aveva laser modeller.

[3] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Gaël Guennebaud, Joshua Levine, Andrei Sharf, and Cláudio Silva. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 03 2016.

[4] Carsten Dachsbacher, Christian Vogelgsang, and Marc Stamminger. Sequential point trees. *ACM Transactions on Graphics*, 22:657, 07 2003.

[5] Jonne Davidsson. Technical report, 3point Oy, Lapinlahdenkatu 16 00180 Helsinki, 1 2019.

[6] Jere Fabritius. Terrestrial three-dimensional laser scanning in aveva pdms. Opinnäytetyö, Tampereen ammattikorkeakoulu, 2009.

[7] Joachim Giesen and Frederic Cazals. Delaunay triangulation based surface reconstruction: Ideas and algorithms. 2006.

[8] Jeff Hecht. Lidar for self-driving cars. *Optics and Photonics News*, 29(1):26–33, Jan 2018.

[9] Hamidreza Houshiar, Jan Elseberg, Dorit Borrmann, and Andreas Nuchter. A study of projections for key point based registration of panoramic terrestrial 3d laser scan. *Geo-spatial Information Science*, 18, 03 2015.

[10] Chien Ming Huang and Yi-Hsing Tseng. Plane fitting methods of lidar point cloud. In *29th Asian Conference on Remote Sensing 2008, ACRS 2008*, 29th Asian Conference on Remote Sensing 2008, ACRS 2008, pages 1925–1930, 12 2008.

[11] Jarmo Huhtanen. *Helsingin sanomat*.

[12] Thomas Kersten and Maren Lindstaedt. Virtual architectural 3d model of the imperial cathedral (kaiserdom) of königslutter, germany through terrestrial laser scanning. 11 2012.

[13] Turner Whitted Marc Levoy. The use of points as a display primitive. Technical report, Computer Science Department, University of North Carolina at Chapel Hill, 1985.

[14] Markus Piipponen. 3d-suunnittelun hyödyntäminen tehdassuunnittelussa. Opinnäytetyö, Satakunnan ammattikorkeakoulu, 2012.

[15] Heikkilä Rauno, Karjalainen Antti, Pulkkinen Pekka, Haapa aho Esa, Jokinen Mauno, Oinonen Aarno, and Jaakkola Mika. Siltojen 3d-suunnittelu- ja mittausprosessin kehittäminen ja käyttöönottaminen (Älykäs silta). Technical report, Tiehallinto, 2005.

[16] Szymon Rusinkiewicz and Marc Levoy. Qsplat: A multiresolution point rendering system for large meshes. *Proceedings of SIGGRAPH*, 2000, 10 2001.

[17] Markus Sch˙ Potree: Rendering large point clouds in web browsers. Master's thesis.

[18] Claus Scheiblauer. *Interactions with Gigantic Point Clouds*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2014.

[19] Pascal Willy Theiler. *Automated Registration of Terrestrial Laser Scanner Point Clouds*. PhD thesis, ETH Zũrich, 2015.

[20] Michael Wimmer and Claus Scheiblauer. Instant points: Fast rendering of unprocessed point clouds. In *Proceedings Symposium on Point-Based Graphics 2006*, pages 129–136. Eurographics, Eurographics Association, July 2006.