Säteenseurantateknologian optimointi reaaliaikaisessa simuloidussa ympäristössä

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tieto- ja viestintätekniikka
Kesäkuu 2024
Kasperi Juutinen

TURUN YLIOPISTO
Tietotekniikan laitos

Kasperi Juutinen: Säteenseurantateknologian optimointi reaaliaikaisessa simuloidussa ympäristössä

TkK-tutkielma, 25 s.
Tieto- ja viestintätekniikka
Kesäkuu 2024

Säteenseuranta on noussut yhdeksi suosituimmista tavoista mallintaa valon toimintaa graafisessa renderöinnissä. Säteenseurantateknologian käyttö on lähivuosina yleistynyt kuluttajatason sovelluksissa. Sen käyttäminen on kuitenkin usein käyttäjien laitteistolle vaativaa. Säteenseurannalle on kehitetty useita optimointimenetelmiä, joilla sen kustannuksia voidaan vähentää. Tämän tutkielman tarkoitus on tarkastella valittuja optimointimenetelmiä, sekä vertailla niitä toisiinsa. Vertailu tehdään, jotta lukija saisi hyvän käsityksen siitä, mihin tekniikoita käytetään ja miten ne eroavat toisistaan. Tutkielmassa on valittu kaksi optimointimenetelmää, jotka ovat kiihdytysrakenteet ja kohinanpoisto. Kiihdytysrakenteiden osalta tutkitaan kolmea valittua rajavolyymihierarkiaa BVH, KD-puu ja CB-puu. Kohinanpoiston menetelmistä käsitellään tilasuodatusta, aikakeräystä ja syväoppimisen käyttämistä.
Tulokset osoittavat, että BVH tai CB-puu on ympäristöstä riippuen tehokkain kiihdytysrakenne reaaliaikaisessa ympäristössä. Kohinanpoistossa koneoppiminen tuottaa parhaan kuvanlaadun, vaikka se onkin laskennallisesti vaativaa. Tutkimuksen perusteella voidaan päätellä, että oikean optimointimenetelmän valinta riippuu käytettävistä resursseista ja suoritusympäristöstä.

Asiasanat: säteenseuranta, reaaliaikainen optimointi, graafinen renderöinti, kiihdytysrakenteet, kohinanpoisto, BVH, KD-puu

Sisällys

Kuvat

1 Johdanto

Renderöinti, eli grafiikkojen hahmontaminen on keskeinen osa tietokonegrafiikkoja. Se tarkoittaa prosessia, jolla luodaan kuva 3D-ympäristöstä. Renderöinti on ollut suosituin ihmisen ja tietokoneen välinen käyttöliittymä, ja todennäköisesti tulee olemaan myös nähtävissä olevassa tulevaisuudessa. Useiden renderöintitekniikoiden keskuudessa realistisella renderöinnillä on ollut keskeinen rooli. Tämä johtuu esimerkiksi korkeatasoisen pelaamisen, elokuvien erikoistehosteiden sekä tietokoneavusteisen suunnittelun suosiosta. [1]

Säteenseuranta on graafisen renderöinnin metodi, jolla simuloidaan valon käyttäytymistä ja vuorovaikutusta eri pintojen kanssa. Realistisen sisällön tuottaminen on ollut tärkeä tavoite useiden eri alojen ammattilaisten keskuudessa. Esimerkiksi videopelien sekä videoiden tuotannossa realistinen valon mallintaminen on ollut tärkeä tekijä laadukkaan sisällön tuottamisessa. Säteenseurannan ollessa tehokas tapa mallintaa todenmukaisesti valoa, sen käytön mukana on esiintynyt myös haasteita. Säteenseurannan käyttö, varsinkin reaaliaikaisessa ympäristössä on kallista käytetyn ajan ja energian perusteella. [2]

Ennen säteenseurannan suosioon nousua yleisesti käytetty renderöintiteknologia oli rasterointi. Algoritmin yleisimmin käytetyt tekniikat kehitettiin 1960- ja 1980-luvun välillä. Rasterointi on vieläkin laajasti käytetty tekniikka renderöinnissä. Tässä metodissa, kohtauksessa esiintyvä esine jaetaan yksinkertaisista geometrisista muodoista muodostuvaksi verkoksi. Nämä muodot sitten lähetetään renderöintipro-

sessiin. Prosessissa muodot jaetaan paloihin, joiden väri sitten selvitetään useiden operaatioiden avulla. Lopuksi jokaisen pikselin varjostus lähetetään kehyspuskuriin tulostusta varten. Suuri ongelma tässä algoritmissa on kuitenkin se, että se noudattaa ”dekonstruktivismiin” perustuvaa lähestymistapaa ja käsittelee muotoja erikseen. Monet optiset renderöintiefektit, kuten varjot, heijastukset ja taittuminen, kuitenkin vaativat useiden muotojen samanaikaista käsittelyä [1]. Tämän takia säteenseuranta nousi suosioon realistisen kuvan tuottamisessa. Säteenseuranta pyrkii mallintamaan valon käyttäytymistä luonnollisesti seuraamalla valon säteitä.

1.1 Tutkielman tarkoitus ja tutkimuskysymykset

Tässä kandidaattitutkielmassa perehdytään säteenseuranta-teknologian optimointiin. Tutkielmassa valittiin tarkasteltavaksi kaksi optimoinnin menetelmää: kiihdytysrakenteet ja kohinanpoisto. Nämä menetelmät valittiin, koska ne ovat vakiintuneita ja laajasti tutkittuja menetelmiä säteenseurannan optimointiin. Tutkielman tavoitteena on tarkastella optimointimenetelmien toimintaa, sekä pohtia, miten ne vaikuttavat tehokkuuteen ja laatuun.

Tutkielmassa tarkastellaan useita tutkimuksia, jotka keskittyvät säteenseurannan optimointimenetelmien toimintaan, sekä uusien menetelmien esittelyyn. Jotta voitaisiin tarkastella optimointimenetelmiä, täytyy tunnistaa, miten säteenseuranta käytännössä toimii. Tutkielmassa selvitetään myös, millaisia menetelmiä voidaan käyttää teknologian optimointiin. Tutkielman tutkimuskysymyksiksi valittiin:

1. Mitä on säteenseuranta, ja miten sen toimintaa voidaan optimoida?
2. Miten valitut optimointitekniikat vaikuttavat reaaliaikaisen säteenseurannan suorituskykyyn ja visuaaliseen laatuun?

1.2 Menetelmät ja tiedonhaku

Tutkielman aineistoja haettiin seuraavista tietokannoista: IEEE, ScienceDirect, ACM ja Web of science. Hakusanoina toimivat ”Ray tracing” AND ”Optimization” AND ”Real-time”, ”Ray tracing” AND ”Optimization” AND ”Real-time” AND ”Acceleration structure” joista hakutuloksia tuli 273. Tuloksista valittiin otsikon ja tiivistelmien perustella 8. Tulosten lisäksi käytettiin connectedpapers.com sivustoa, johon syötettiin saatuja aiheeseen liittyviä papereita. Sivuston avulla löydettiin 9 aiheeseen liittyvää tutkimusta. Tutkielmaan löytyi myös muutama lähde tutkimalla aikaisempien tulosten lähdeluetteloita. Tietoa tutkielmaan saatiin myös developer.nvidia.com nettisivulta. Nvidia on tunnettu säteenseurantaa hyödyntävä yritys, jonka nettisivut todettiin luotettavaksi lähteeksi.

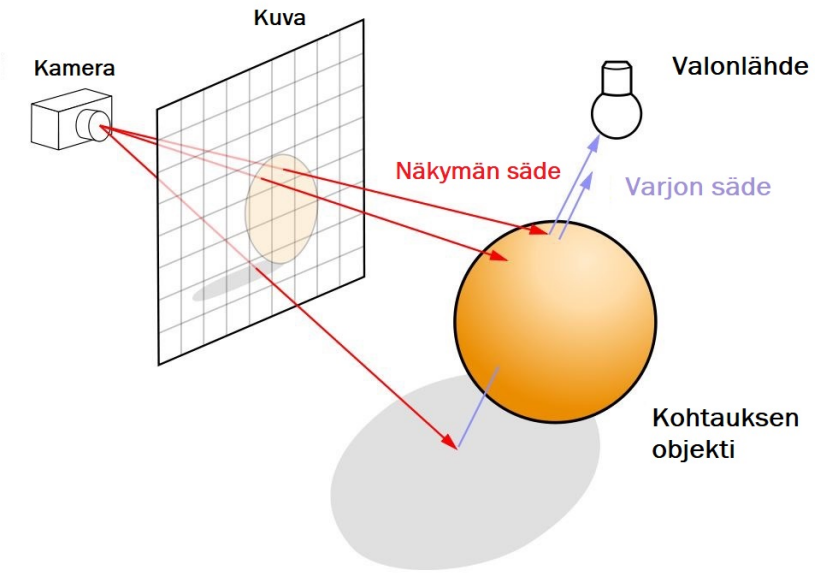
1.3 Lukujen jäsennys

Tutkielma koostuu johdannon lisäksi viidestä muusta luvusta. Luvussa 2 käydään läpi säteenseurantaa konseptina, sekä avataan sen toimintaa matemaattisesti. Tässä luvussa esitetään myös säteenseurantaan käytettäviä algoritmejä. Luvussa 3 käydään läpi säteenseurannassa esiintyvät yleisimmät haasteet. Luvussa 4 käsitellään säteenseurannan optimointia ja optimointimenetelmiä. Tässä luvussa esitellään ja perehdytään valittuihin optimointimenetelmiin. Luvussa 5 vertaillaan valittuja optimointimenetelmiä toisiinsa ja luodaan johtopäätöksiä vertailun perusteella. Luvussa 6 luodaan yhteenveto aikaisempien lukujen johtopäätöksistä ja tuloksista. Tässä luvussa pohditaan tutkielman tuloksia ja pohditaan mahdollista jatkotutkimusta aiheelle.

2 Säteenseurannan toiminta

2.1 Säteenseuranta konseptina

Säteenseuranta on graafisen renderöinnin tekniikka, joka voi realistisesti simuloida kohtauksen, ja sen esineiden valaistusta tekemällä fyysisesti tarkkoja heijastuksia, taittumia, varjoja ja epäsuoraa valaistusta. Renderöinti tarkoittaa tietokonegrafiikoissa käytettävää prosessia, jossa luodaan kuvia kolmiulotteisesta kohtauksesta. Säteenseuranta generoi tietokonegrafiikka-kuvia seuraamalla valon polkua näkymän kamerasta kaksiulotteisen näkymäpinnan läpi kolmiulotteiseen kohtaukseen, josta sitten seurataan valoa valonlähteisiin. Kun valo kulkee kohtauksessa, se voi heijastua esineiden välillä, tulla estetyksi esineiden toimesta (aiheuttaen varjoja) tai kulkea läpinäkyvien tai osittain läpinäkyvien esineiden läpi (aiheuttaen taittumista). Nämä interaktiot sitten yhdistetään luomaan näytöllä näytettävän pikselin väri ja valaistus. [1] [3] Kuvassa 2.1 esitetään yksinkertaistettuna säteenseuranta-algoritmin toiminta.



Kuva 2.1: Kaavio havainnollistaa kuvan renderöimiseen käytettävän säteenseuranta-algoritmin toiminnan. Alkuperäinen kuva on lisensoitu CC BY-SA 4.0 lisenssillä. [4]

Renderöinnissä on valittu valolähteestä tulevien säteiden seurannan sijasta säteiden seuraaminen kamerasta valonlähteisiin. Tämä käänteisen prosessin valinta perustuu siihen, että valonsäteiden seuraaminen kamerasta on paljon tehokkaampaa, kuin valonlähteistä lähtevien säteiden seuraaminen useisiin suuntiin. [3] Kun säteitä seurataan kamerasta, huomioon ei tarvitse ottaa säteitä, jotka eivät osu kameraan.

Säteenseurannassa termi säteensuuntaus tarkoittaa prosessia, jossa säteenseuranta-algoritmi lähettää yhden tai useamman säteen kamerasta jokaisen kuvatason pikselin läpi ja testaa sitten, leikkaavatko säteet kohtauksessa sijaitsevia primitiivejä. [3] Primitiivit tarkoittavat muotoja, joihin säteenkohtaamista käsittelevä rutiini on kirjoitettu. Kohtauksen käsiteltävät esineet ovat usein mallinnettu kolmion muotoisten primitiivien verkoksi, jotta säteenkohtaamisen käsittelyä voidaan yksinkertaistaa. Kuvassa 2.2 esitetään säteenseurannalla ja rasterisaatiolla renderöity kohtaus. Kuvan tarkoitus on esittää visuaalisia eroja tekniikoiden välillä. Säteenseurannalla tuotetussa kuvassa nähdään valon heijastuvan todenmukaisesti teepannun ja lusi-

kan pinnasta. Säteenseurannassa nähdään valon heijastumisen lisäksi esineiden heijastuminen peilin kaltaisista pinnoista. Varjojen tarkkuus ja laatu totetuvat myös todenmukaisemmin säteenseurannalla toteutetussa kuvassa.

Kuva 2.2: Kuvassa esitetään kaksi lopputulosta saman kohtauksen renderöinnistä. Toisessa on käytetty rasterointia ja toisessa säteenseurantaa. Kuva on lisensoitu CC BY 4.0 lisenssillä. [5]

2.2 Säteenseurannan matemaattinen malli

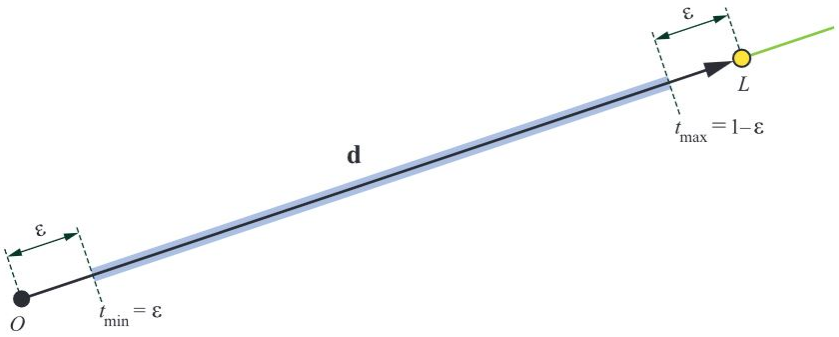
Tärkeä laskennallinen rakenne säteenseurannassa on yksittäinen kolmiulotteinen säde. Matematiikassa ja säteenseurannassa säde viittaa yleensä kolmiulotteiseen puolisuoraan. Sädettä kuvataan tavallisesti parametrisessä muodossa. Parametrinen suora voidaan esittää pisteiden A ja B painotettuna keskiarvona seuraavalla yhtälöllä:

$$P(t) = (1 - t)A + tB$$

Parametri t voi tässä tapauksessa saada minkä tahansa reaaliluvun arvon, eli $t \in [-\infty, +\infty]$. Piste P liikkuu siis yhtälössä jatkuvasti pitkin suoraa arvon t muuttuessa.

Funktion implementointiin tarvitaan suoralta pisteet A ja B. Ohjelmointirajapinnoissa (API) ja ohjelmointikielissä esitystapaa kutsutaan usein nimellä ”vec3” tai ”float3”, ja se sisältää kolme reaalilukua x, y ja z. Pisteet A ja B voidaan valita mistä

vain kahdesta eri kohdasta suoralla. Säteen kanssa tulee usein lisänä väli: arvojen t joukko, jolloin leikkauspiste on hyödyllinen. Väli esitetään yleensä kahtena arvona t_{min} ja t_{max}, jotka siis rajaavat arvon $t \in [t_{min}, t_{max}]$. Tämä tarkoittaa siis sitä, että jos leikkauspiste löydetään arvosta $t < t_{min}$ tai $t > t_{max}$, sitä leikkausta ei huomioida. Kun leikkauspiste on renderöinnin kannalta merkityksettömällä etäisyydellä, sille annetaan maksimiarvo. Tällaista tilannetta käytetään esimerkiksi varjosäteiden mallintamisessa. [6] Kuvassa 2.3 esitetään säteen puolisuoran toiminta.



Kuva 2.3: Kuvaaja esittää esimerkin säteen puolisuorasta, jossa valonlähde on pisteessä L ja lähtöpiste kohdassa O. Suoralla etsitään leikkauspisteitä pisteiden O ja L välillä. Säteen väliä $[t_{min}, t_{max}]$ käytetään rajoittamaan leikkauspisteiden etsintää. Tarkkuusongelmien välttämiseksi rajoitus toteutetaan asettamalla säteen väliksi $[\epsilon, 1 - \epsilon]$. Täten saadaan siis kuvaajassa sinisellä esitetty tarkasteluväli **d**. CC Attribution-NonCommercial-NoDerivatives 4.0 International License. [6]

Oletetaan, että varjostamme pistettä P ja haluamme tarkistaa valon näkyvyyden kohdassa L. Luodaan säde, jonka lähtöpiste on $O = P$ ja suuntavektori $d = L - P, t_{min} = 0$ ja $t_{max} = 1$. Jos leikkaus tapahtuu, kun arvo t on välillä [0, 1], säde leikkaa valoa peittävää geometriaa.

2.3 Algoritmit

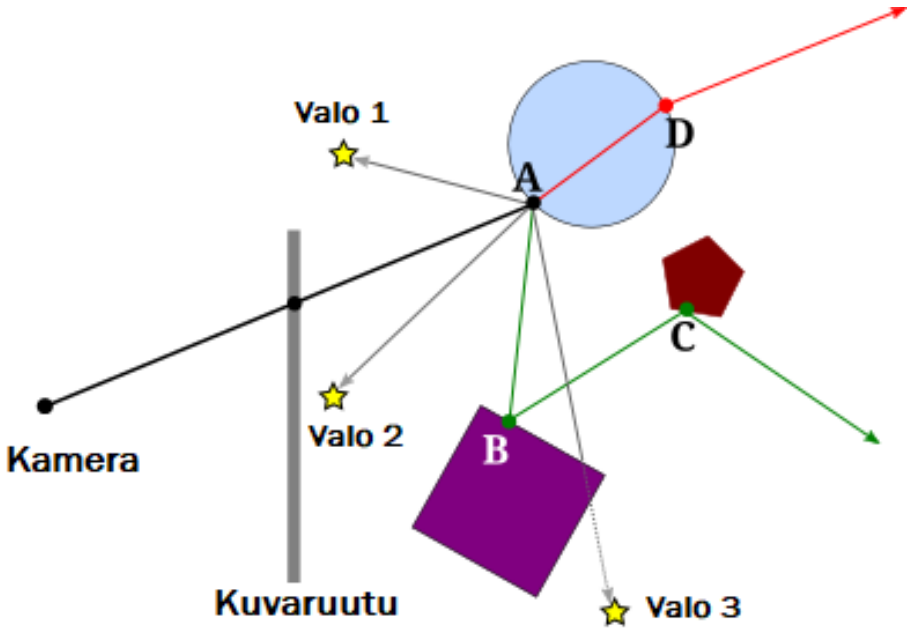
Säteenseurannalle on muodostettu useita algoritmeja erilaisia tarkoituksia varten. Perinteisen säteenseuranta-algoritmin ehdotti ensimmäisen kerran Appel vuonna 1968 [1]. Alkuperäinen versio tunnettiin säteidenheittoalgoritmina, sillä se heijasti

säteitä kamerasta ja tarkisti sen jälkeen leikkauspisteet. Tämä tekniikka mahdollisti kuitenkin vain karkeasti arvioidun kuvan näyttämisen. 1980-luvulla Turner Whitted esitteli säteenseuranta-algoritmin, joka pystyi käsittelemään monimutkaisempia valaistusvaikutuksia. [7] Whitted-säteenseuranta-algoritmi on nykyään pääasiallinen säteenseuranta-algoritmi reaaliaikaisille sovelluksille.

Whitted-algoritmissa säteen osuessa kohtauksessa sijaitsevaan primitiiviin, se voi myös luoda sekundäärisäteitä, joihin kuuluu heijastus-, taittumis- ja varjosäteet. Säteitä voidaan sitten toistaa rekursiivisesti. [1] Kuva 2.4 esittää Whitted-algoritmin säteenseurannan toimintaa yhdelle pikselille. Kuvassa esiintyy kolme valonlähdettä, sekä kolme esinettä, joista yksi on osittain läpinäkyvä. Vihreä suora esittää osittain läpinäkyvän esineen pisteestä A heijastunutta sädettä, joka osuu toisen esineen pisteeseen B. Vihreän suoran avulla katsoja näkee pisteen B heijastuksen pisteessä A. Tämä otetaan huomioon, koska esineet, joilla on peilin kaltainen pinta, heijastavat valon lisäksi myös muita esineitä. Punainen suora esittää esineen läpinäkyvyyttä ja säteen taittumista esineen sisällä.

Algoritmi ottaa syötteeksi kohtauksessa generoidun primitiivien joukon. Heijastetaan sitten kamerasta säde jokaista ruudun pikseliä kohti. Nämä säteet on määritetty pääsäteiksi. Algoritmi käy sitten läpi kaikki primitiivit, ja tunnistaa niistä katsojaa lähimmän. Kun lähin primitiivi on löydetty, sekundäärisäteitä voidaan luoda ja seurata. Suoraviivaisella toteutuksella algoritmi on O(NM)-kompleksinen, jossa N vastaa säteiden määrää ja M primitiivien määrää. [1]

Rekursiivisessa säteenseurannassa uusia säteitä voitaisiin muodostaa ääretön määrä. Tämä ei kuitenkaan ole realistista säteenseurannan toteutuksissa. Yksi tapa rajoittaa uusia säteitä, on määritellä rekursioille tietty syvyys. Yleisesti käytetty syvyys on viiden rekursion pituinen. Parempi ratkaisu tähän voisi kuitenkin olla dynaaminen syvyys. Heijastus- ja taittumiskertoimet ovat säteenseurannassa käytettyjä arvoja. Kertoimien arvot ovat välillä [0,1]. Heijastuskerroin määrittää, kuinka



Kuva 2.4: Kuvaaja esittää Whitted-säteenseuranta-algoritmin rekursiivisen toiminnan. Kuvaaja esittää yhdelle pikselille lähetetyn yhden säteen toimintaa. Kuvaajassa esiintyy kolme valonlähdettä sekä kolme esinettä, joista yksi on osittain läpinäkyvä. Suorat esittävät säteiden toimintaa algoritmissa. Creative Commons Attribution-Noncommercial-ShareAlike 4.0 License. [8]

suuri osa valosta heijastuu pinnasta. Taittumiskerroin taas määrittää, kuinka suuri osa valosta taittuu pinnan läpi. Koska heijastus- ja taittumiskertoimet ovat välillä [0,1], säteiden etenemisen yhteydessä valaistusmäärien kertominen näillä kertoimilla aiheuttaa nopean valaistusvaikutuksen vähenemisen syvemmillä rekursiotasoilla. Tämä tarkoittaa siis sitä, että syvempää rekursiota voidaan välttää ilman huomattavaa vaikutusta lopulliseen kuvaan. [9]

Säteenseurannassa toinen yleisesti käytetty algoritmi on polunjäljitysalgoritmi. Polunjäljitysalgoritmia käytetään globaalin valaistuksen ja fotorealististen kuvien laskemiseen. [10] Polunjäljitysalgoritmi voidaan toteuttaa lähettämällä säteitä satunnaisiin suuntiin globaalien valaistusvaikutusten aikaansaamiseksi. Säteiden määrän täytyy olla riittävän suuri, jotta hyvä stokastinen arvio voidaan selvittää. [1]

2.4 Fysikaalisen valon ja näön vaikutukset säteenseurantaan

Valo on sähkömagneettista säteilyä, jota voidaan havaita silmällä. Kun valo osuu esineeseen, se voi heijastua, taittua tai absorptoitua, riippuen esineen koostumuksesta. Heijastuminen tarkoittaa valon kimpoamista esineestä. Peilit ja peilin kaltaiset pinnat heijastavat melkein kaiken tulleen valon. Valon taittuminen tarkoittaa valosäteen suunnan muuttumista esineen sisällä. Taittumista voidaan havaita esimerkiksi säteen kulkiessa veden läpi, jolloin valon suunta muuttuu. Valon absorptio tarkoittaa säteiden imeytymistä pintaan. Pinnan tummuus ja karkeus vaikuttavat siihen, kuinka paljon valoa pinta absorboi. [11]

Ihmissilmällä on useita ainutlaatuisia ominaisuuksia, joilla on vaikutus resoluutiota tarkastellessa. Nähty resoluutio tippuu nopeasti, kun liikutaan pois alueelta, johon silmä on keskittynyt. Peuhkarinen ja Mikkonen [12] toteavat, että edellä mainitun huomioon ottaen renderöintijärjestelmä voitaisiin suunnitella siten, että renderöinnin laatu ei ole yhtä korkea reuna-alueella, kuin alueella, jota parhaillaan katsotaan, ilman että ihminen huomaisi. [12]

Mostafawy ym. [13] esittelivät ensimmäisen näköä simuloivan säteenseurantametodin. Heidän metodissaan simuloitu kuva luodaan keskimääräistämällä säteiden joukon tulokset, jotka jäljitetään optisen järjestelmän läpi. Tämä prosessi suoritetaan sitten jokaiselle pikselille. Tämä menetelmä on rakennettu Gullstrandin silmämallille [13]. Malli käyttää pallopintoja mallintamaan silmän taittavia elementtejä.

Tässä metodissa kuvatason oletetaan olevan tasainen. Tämä ei siis kykene aidosti simuloimaan silmän kaarevia verkkokalvoja. Tämä johtaa esimerkiksi siihen, että simulaatioiden reuna-alueet ovat liian sumeat. [14]

3 Haasteet

Säteenseuranta on suoraviivainen ja tehokas tekniikka, jota voidaan käyttää heijastusten ja varjojen toteuttamiseksi. Sen laskennallinen vaativuus on kuitenkin usein liian suuri efektien reaaliaikaiseen implementointiin nykyaikaisilla grafiikkaprosessoreilla. Tästä syystä on turvauduttu vaihtoehtoisiin menetelmiin efektien toteuttamiseksi. [15]

Säteenseuranta tunnetaan useiden muiden ominaisuuksien lisäksi sen hienostuneisuudesta ja yksinkertaisuudesta. Uusia renderöintialgoritmeja ja -efektejä voidaan lisätä vain jäljittämällä joitakin säteitä. Uusia pinta-primitiivejä voidaan lisätä yksinkertaisesti vain määrittelemällä niiden rajauslaatikko ja leikkauksenkäsittelyohjelmat. On kuitenkin useita tapauksia, missä nämä väitteet eivät päde. Esimerkiksi tapaukset, joissa oletusarvoisen leikkausluvun etsimiseen käytettävä rajapinta ei enää riitä; missä rajallisen liukuluvun tarkkuus pilaa matemaattiset laskut; missä "reunatapaukset", kuten useat samassa tasossa olevat pinnat, pienet tai kaukaiset geometriat, tai suuresti vaihtelevat kustannukset pikseliä kohden, hankaloittavat tilannetta. Esimerkki ongelmatilanteesta on itseleikkaus, joka voi tapahtua, kun säde jatkaa kulkua leikkauksen jälkeen ja osuu uudelleen samaan pintaan. Säde siis tulkitsee leikkaustilanteen uudelleen saman pinnan kanssa. Toteutuksessa usein käytetty rajallinen liukulukutarkkuus johtaa usein vääriin positiivisiin tuloksiin. Haasteita voisi olla houkutteleva sivuuttaa harvinaisina tapauksina, mutta käytännössä sivuuttamisen mukana tulisi riskejä. [6] Jos harvinaiset tapaukset jätettäisiin huo-

mioimatta, ne voisivat esiintyä suoritustilanteessa, aiheuttaen ennalta arvaamatonta käyttäytymistä valolle.

Säteenseurannan optimoinnissa on tyypillisesti keskitytty kuvaruutujen renderöintiin kuluvan ajan vähentämiseen. Nykyaikaisissa tietokoneissa saattaa olla kuitenkin tärkeämpää vähentää renderöintiin käytettyä energiaa. Energiarajoitteiset arkkitehtuurit, kuten mobiiliympäristöt, kasvattavat energiankulutuksen ja renderöintiajan vähentämisen ja tutkimisen tärkeyttä. Nykyaikaisissa arkkitehtuureissa ajan ja energian vaatimukset korreloivat vahvasti tiedonsiirron kanssa. Vaikka säteet vaikuttavatkin itsenäisesti lopulliseen kuvaan, datan liikkumisen suorituskyky on riippuvainen muistijärjestelmän kokonaistilasta. Tämä tarkoittaa siis sitä, ettei suorituskyvyn laskemisessa voida pelkästään ottaa huomioon suoritettavien käskyjen määrä. Suorituskyvyn tarkastelemisessa täytyy siis huomioida tiedon liikkuminen koko renderöintiprosessissa. [2] Mitä syvemmälle rakenteeseen säteitä seurataan, sitä suurempi osa kohtauksesta tulee ottaa huomioon ja siten ladata.

Säteenseuranta skaalautuu hyvin, kun käytettävissä oleva laskenta lisääntyy. Muistijärjestelmästä kuitenkin nopeasti muodostuu pullonkaula, kun säteenseurantaalgoritmit kuormittavat sitä satunnaisella muistinkäytöllään. Tämän lisäksi vapaana olevan laskentatehon lisääntyessä energiankulutus lisääntyy. Uudeksi ongelmaksi tehokkaissa piireissä onkin esiintymässä niiden käyttämä teho ja lämmöntuotto, jotka rajoittavat, kuinka suurta osaa piiristä voidaan pitää aktiivisena samaan aikaan. [16]

4 Optimointi ja optimointimenetelmät

4.1 Optimointi

Tehokkuuden parantaminen on ollut nykytietotekniikassa keskeinen aihe. Säteenseurannan suorituskyvyllä on kuitenkin teoreettinen yläraja, sillä sen suorituskyky ja tehokkuus eivät skaalaudu lineaarisesti laskentaresurssien lisääntyessä. Tämän rajan takia optimisaatioita on ehdotettu algoritmeihin. Optimoinnilla on pyritty parantamaan suorituskykyä sekä vähentämään muistiliikennettä ja tiedonsiirtoa. Kaistanleveyden käyttö on tunnistettu suurimmaksi pullonkaulaksi perinteisessä säteenseurannassa. [2]

Elena Vasiou ym. [2] tarkastelivat, energian ja ajan kulutusta renderöinnin aikana. Tutkimuksen suorittamiseksi he simuloivat renderöintiä TRaX-arkkitehtuurilla. TRaX-arkkitehtuuri on säteenseurantaan keskittyvä laitteistoarkkitehtuuri. Tutkimuksessa suoritetaan 50 simulaatiota, joiden tuloksia tarkastellaan ja luodaan johtopäätöksiä. Testin tulokset osoittavat, ettei renderöinnin ajan ja energian kulutuksen parannuksia voi tehdä yksinkertaisesti lisäämällä käytettävissä olevia muistiresursseja. Tutkimuksessa muistijärjestelmä tunnistettiin energian ja ajan kulutuksen päälähteeksi. Havaintojen perusteella suositellaan harkittavaksi kokonaisvaltaisempia suorituskyvyn optimointeja funktiona renderöintiajasta, energiankulutuksesta

sekä säteiden vaikutuksesta kuvan laatuun. [2]

Suosittu tapa ohjelmiston optimointiin on jakaa käsiteltyä tilaa. Tämä auttaa vähentämään laskennassa verrattavien kohteiden määrää. Tällaiset tekniikat käyttävät yleensä jonkinlaista tietorakennetta, ja todennäköisesti käyttävät puumaista tietorakennetta. Toinen yleinen optimointimenetelmä on säteiden ryhmittely. Säteitä voidaan suunnata ryhminä yksittäisten säteiden sijaan, sillä kaikilla säteillä on sama lähtöpiste. Viereiset säteet myös usein käyttäytyvät samalla tavalla. [10]

4.2 Kiihdytysrakenteet

Säteenseurannan optimoinnissa yksi tärkeimmistä tavoista vähentää kompleksisuutta on käyttää kiihdytysrakenteita. Ilman kiihdytysrakenteita, jokainen primitiivi tarkistetaan jokaiselle säteelle. Kiihdytysrakenteet järjestävät primitiivit hierarkiseen spatiaaliseen rakenteeseen sopeutuen niiden jakautumiseen. Rakenteella voidaan tunnistaa säteiden läheisimmät leikkauspisteet. [1] Rakenteiden avulla kompleksisuutta voidaan vähentää lineaarisesta logaritmiseksi säteen leikkauspisteiden määrän suhteen [6].

4.2.1 Rajavolyymihierarkia

Yksi eniten käytetyistä kiihdytysrakenteista säteenseurannan optimointiin on rajavolyymihierarkia eli BVH (Bounding Volume Hierarchy). BVH tarkoittaa yksinkertaisesti rajaavista volyymeistä muodostuvaa puuta. Puurakenteessa jokainen solmu on yhdistetty rajoittavaan laatikkoon, joka on määritetty akseliin suuntautuvaksi rajoittavaksi laatikoksi [1]. Akselilla tarkoitetaan tässä tilanteessa suuntaviivaa, jonka mukaan tila jaetaan. Rajaavan volyymin solmu sulkee sen lasten rajaavat volyymit. Jos säde ei osu rajaavan volyymin tiettyyn solmuun, säde ei myöskään osu mihinkään tämän solmun lapsista. Tämä tarkoittaa siis sitä, että laskiessa kaikki solmun lapset voidaan ohittaa. Säteenheittoalgoritmi kulkee hierarkian läpi, yleensä

syvyyssuuntaisessa järjestyksessä, ja määrittää, leikkaako säde esineen.

BVH-rakenteita voidaan rakentaa usealla tavalla. Helpoin tapa on rakentaa BVH rekursiivisesti. Rakenteessa BVH ottaa listan primitiivejä sisältävistä rajaavista volyymeistä, ja järjestää ne akselin mukaan. Lista jaetaan sitten kahtia, kummallekin puoliskolle asetetaan rajoittava laatikko, ja akseleita käydään läpi rekursiivisesti. Koska rakenne käyttää akselin mukaisia rajaavia laatikoita, se ei ole ihanteellinen rakenne ympyrän muotoisten esineiden käsittelemiseen. [17]

BVH:t tarjoavat tehokkaan tavan organisoida kolmiulotteisen kohtauksen spatiaalista jakautumista. Niillä voidaan nopeasti määrittää, mihin osiin kohtausta säde voi mahdollisesti osua. Tehokkaimmat BVH -metodit käyttävät pinta-alueen heuristiikkaa BVH:ita muodostaessa. Tämä auttaa määrittämään tehokkaan hierarkisen esineiden jakamisen. Tämä metodi antaa kuitenkin vain staattisen suuntauksen, ja ei ota huomioon myöhempien sädekyselyiden luonnetta. [18]

4.2.2 KD-puu

KD-puu, eli k-ulotteinen puu on erikoistapaus binäärisestä tilanjako-puusta. KD-puu ehdotettiin alunperin tilaa rajoittavana datarakenteena, jonka tarkoitus oli indeksoida pisteitä k-ulotteisessa tilassa [19]. KD-puu rakennetaan rekursiivisesti jakamalla tilaa kahteen puoliskoon annetun akselin suuntaisesti. Leikkaus valitaan niin, että kummallakin puoliskolla on suurinpiirtein yhtä monta esinettä. Säteenseurannan käsittelyssä esineiden jakaminen tasaisesti ei kuitenkaan usein ole ihanteellinen strategia. Alueiden välillä geometrinen kompleksisuus saattaa vaihdella, esimerkiksi pienten yksityiskohtien myötä. Tietojen tasainen jakaminen ei myöskään ota huomioon esineiden todellista muotoa tai tiheyttä, joka saattaa johtaa epätasapainoiseen puuhun. Parempi ratkaisu voitaisiin saada ottamalla huomioon primitiivien määrän sekä satunnaisen säteen mahdollisuuden osua jompaankumpaan puolitetuista alueista. Näiden laskujen tekeminen on suhteellisen hintavaa, jonka takia niissä käytetään yleensä erilaisia likimääräisiä ratkaisuja.

Kun primitiivit on järjestelty KD-puuksi, säteenseuranta-ohjelma suorittaa kulkuoperaation jokaiselle säteelle. Pääsolmusta alkaen ohjelma tarkistaa rekursiivisesti, miten sisääntulevat säteet leikkaavat tilan kanssa. Koska kumpaankin lapsisolmuun liittyvät tilat voidaan leikata, tässä käytetään pinoa tallettamaan liittyvät solmut. Pino tarkoittaa tietorakennetta, johon tallennetaan solmut, jotta niitä voidaan käsitellä tehokkaasti ja oikeassa järjestyksessä. Kun KD-puun lehtisolmuun saavutaan, ohjelmassa iteroidaan solmun sisältämät primitiivit, jotta saadaan selville lähin säteen leikkaama primitiivi. [1]

4.2.3 CB-puu

Kowshik ym. [10] esittelivät uuden kartioihin perustuvan kiihdytysrakenteen. Työn tarkoitus on vähentää ajan kompleksisuutta. Tutkielman esittelemä CB-puu hyödyntää kartioita ryhmittelemään säteitä, mikä mahdollistaa säteiden ja objektien leikkauspisteiden välisen tehokkaamman laskennan.

Kartiopohjainen puu pakkaa joitakin säteitä ryhmiin käyttämällä kartioita, joiden avulla voidaan tunnistaa kaikki mahdolliset esineet, jotka voivat leikata säteiden kanssa. Leikkaavat esineet määritetään aktiivisiksi esineiksi. Tämä rakenne on suunniteltu vähentämään tarpeettomien leikkauslaskentojen määrää, nopeuttaen renderöintiä ja säilyttäen kuvan yksityiskohtaisuuden.

CB-puita muodostetaan pääsäteiden ja varjosäteiden käsittelyyn. Näiden säteiden osalta käytetään eri määritelmää kartion laskemiseen aktiivisten objektien osalta. Ensisäteille suunnattua puuta kutsutaan pääsäteille suunnatuksi puuksi (CPR), ja, varjosäteille suunnattua puuta varjosäteille suunnatuksi puuksi (CSR). CPR-puu toimii varastoimalla ensisäteistä muodostuvien kartioiden aktiiviset esineet puurakenteeseen. CSR-puu eroaa siten CPR-puusta siten, että varjosäteet muodostuvat sekä kovista, että pehmeistä varjoista. Kova varjo tarkoittaa varjoa, jossa esineestä täytyy suunnata säteitä valonlähteeseen. Pehmeässä varjossa valo on määritelty alueena, eikä tiettynä pisteenä. Algoritmin täytyy siis laskea aktiiviset esineet mo-

lemmille varjotyypeille.

Kowshik ym. [10] ehdottavat uudenlaista CB-puu-menetelmää, joka yhdistää pääsäteiden ja varjosäteiden laskemiseen käytettäviä kartioita. Metodi sisältää siis CPR- ja CSR-puut. Tutkimuksessa metodin havaittiin toimivan ajankäytön suhteen tehokkaammin pehmeiden ja kovien varjojen seurannassa, kuin pelkästään CPR- tai CSR-puun käyttäminen. Tutkimuksessa saatiin vähennettyä säteiden leikkauspisteiden määrää huomattavasti. Tässä vertailtiin kuitenkin vain perinteistä säteenseurantaa, joten muiden algoritmien suorituskyvystä ei ole varmaa tietoa. [10]

4.3 Kohinanpoisto

Yksi toinen usein käytetyistä optimointimenetelmistä on kohinanpoisto. Sen tarkoitus on parantaa renderöityjen kuvien laatua [14]. Kuvan kohinanpoisto on menetelmä, jolla poistetaan kohinaa meluisasta kuvasta, jotta saataisiin palautettua alkuperäinen kuva. Säteenseurannassa kohina tarkoittaa kirkkautta ja värejä, jotka eivät esiinny alkuperäisessä kuvassa. Kohinaa muodostuu kuvaan ympäristön, siirtokanavien ja muiden tekijöiden vaikutuksesta kuvan ottamisen, pakkaamisen ja siirron aikana. Kohina, reunat ja tekstuurit ovat korkean taajuuden komponentteja, joten niitä on vaikea erottaa toisistaan kohinaa poistaessa. Korkean taajuuden komponentti tarkoittaa kuvan alueita, joissa kirkkausarvot muuttuvat nopeasti, kuten reunat ja yksityiskohtaiset tekstuurit. Tämä voi johtaa siihen, että kuvat menettävät joitakin yksityiskohtia. [20]

Kuvan yksityiskohtien säilyttämiseksi kohinanpoistossa tulee huomioida kolme eri valosignaalia: hajanainen valo, joka heijastuu kaikkiin suuntiin, heijastava valo, joka heijastaa tiettyyn suuntaan sekä äärettömän valolähteen varjot, kuten auringonvalo ja muiden valonlähteiden muodostamat varjot. Kohinanpoisto muodostuu yleisesti kolmesta tekniikasta: tilasuodatus, aikakeräys ja syväoppimisen avulla tapahtuva kuvan uudelleen rakennus.

Tilasuodatus tarkoittaa valikoivasti tapahtuvaa kuvan pikselien muokkausta lähellä sijaitsevien pikselien avulla. Tilasuodatus pystyy vastaamaan tilan muutoksiin nopeasti, se kuitenkin aiheuttaa myös kuvan värähtelyä ja visuaalisia epätäydellisyyksiä.

Aikakeräys tarkoittaa tiedon kierrätystä edellisestä kuvaruudusta. Sen tarkoitus on määrittää, onko kyseisessä kuvassa visuaalisia poikkeamia tai artifakteja, joita voidaan korjata. Tämä metodi aiheuttaa hieman aikaviivettä, mutta se ei aiheuta sumeutta ja vähentää kuvan värähtelyä.

Koneoppimista voidaan käyttää vähentämään kuvan kohinaa rakentamalla kuva uudelleen. Vaikka uudelleen rakennettu kuva voi näyttää valmiilta, tekniikka voi aiheuttaa ajallista epävakautta kuvien välillä. [21]

5 Vertailu

Tutkielmassaan Havran suosittelee KD-puuta staattisen kohtauksen luomiseen [22]. Rakenteen suorituskyky staattisessa rakenteessa on keskiarvoltaan parempi, kuin muiden kiihdytysrakenteiden, kuten BVH:n. Valopuun toteuttamisessa Bikker [23] käytti artikkelissaan kuitenkin BVH:ta. Valopuu tarkoittaa tiettyyn pisteeseen kohdistuvista valoista muodostettua rakennetta, jota käytetään valojen sijainnin hallintaan. BVH valittiin, koska satojen valojen laskeminen KD-puulla olisi epäkäytännöllistä. BVH rakennetaan kerran jokaisessa kuvaruudussa valojen sijaintien päivittämiseksi. Leikkauspisteeseen vaikuttavat valot voidaan BVH:ta läpikäymällä määrittää nopeasti. BVH:n käyttö ratkaisi ongelmia jaotteluheuristiikan kanssa. Joskus valot menevät merkittävästi päällekkäin, jolloin hyvän jaottelutason löytäminen on vaikeaa. KD-puun suorituskyky kärsii monimutkaisesta läpikäyntialgoritmista, jos käsiteltävänä on säieohjelma. KD-puu hyvittää tätä kuitenkin jonkin verran sen kyvyllä sopeutua geometrisen tiheyden muutoksiin. [23]

Thrane ja Simonsen [24] totesivat BVH:n suoriutuvan jatkuvasti paremmin, kuin KD-puun. He totesivat, että BVH oli heidän tutkimushetkellään nopein kiihdytysrakenne säteenseurantaan grafiikkasuorittimessa. Artikkeli on kuitenkin julkaistu vuonna 2005, joten tämä väite ei välttämättä ole täysin totuudenmukainen nykypäivänä. Heidän mukaansa BVH:n rakennuksen ja kulun toteuttaminen on myös yksinkertaisempaa kuin muiden rakenteiden. [24] Deng ym. [1] huomauttavat artikkelissaan, että on vieläkin avoin kysymys, mahdollistaako KD-puu vai BVH kor-

keamman tason läpikäyntitehokkuuden. BVH:n luontainen ongelma on se, että kaksi samalla tasolla sijaitsevaa solmua saattavat olla päällekkäin tilassa, johtaen suurempaan määrään läpikäytäviä solmuja ennen osuman paikantamista. KD-puussa ei ole samanlaista ongelmaa, mutta se käyttää enemmän askelia primitiivien käsittelyyn. [1]

Kowshik ym. [10] suorittavat tutkielmassaan testejä CB-puun suorituskyvystä. He toteuttivat metodia hyödyntäen perinteistä säteenseuranta-algoritmia, OpenGL-kirjastoa sekä C++ kieltä. Testissä tarkastellaan leikkauspisteiden määrää ajan sijasta, sillä renderöintiin kuluva aika vaihtelee laitteen perusteella. Testeissä verrattiin CPR-puuta, CSR-puuta, CB-puuta sekä perinteistä säteenseurantaa. Taulukoissa 5.1 ja 5.2 kuvataan säteiden määrää esineiden määrän suhteen. Taulukon suhdeluvut on saatu jakamalla metodin leikkauspisteiden määrä perinteisen algoritmin leikkauspisteden määrällä. Luvut, jotka ovat alle 1 tarkoittavat siis käsiteltävien leikkauspisteiden määrän vähenemistä peristeiseen säteenseurantaan verrattuna.

Taulukko 5.1: Kovien varjojen vertailu esineiden määrän suhteen

	Esineiden määrä				
	200	**400**	**600**	**800**	**1000**
CPR-puu	0.627	0.669	0.68	0.68	0.671
CSR-puu	0.483	0.382	0.354	0.346	0.351
CB-puu	0.11	0.052	0.034	0.025	0.021
Perinteinen	1	1	1	1	1

Taulukko 5.2: Pehmeiden varjojen vertailu esineiden määrän suhteen

	Esineiden määrä				
	200	**400**	**600**	**800**	**1000**
CPR-puu	0.982	0.984	0.985	0.985	0.984
CSR-puu	0.041	0.029	0.026	0.023	0.022
CB-puu	0.023	0.013	0.011	0.008	0.007
Perinteinen	1	1	1	1	1

Tulosten perusteella CB-puun suorituskyky on paljon parempi, kuin perinteisen säteenseurannan. CB-puulla saavutetaan myös parempia tuloksia, kuin pelkällä

CSR- tai CPR-puulla. Tutkimuksessa tullaan siihen tulokseen, että staattisen kuvan käsittelyssä CB-puulla on paremmat suorituskyvyn arvot perinteiseen säteenseurantaan verrattuna. CB-puu tulee kuitenkin luoda uudelleen, jos kohtauksen esineissä, valoissa tai kamerassa esiintyy animaatiota. CB-puu hyödyntää CPR- ja CSR-puita, joten sen suorittamisessa ilmenee molempien puiden kulut. CPR-puussa kulut koostuvat käsiteltävien pikselien ja esineiden määrästä. Siihen sisältyy siis kartioiden ja esineiden leikkauksien laskut. CSR-puun kulut riippuvat kohtauksen esineiden ja valonlähteiden määrästä. Siinä lasketaan esineiden ja valon määrittelemien kartioiden leikkauspisteitä. Testissä esineiden suurin käytetty määrä oli 1000. Suorituskyvystä suuremmalla esine- tai pikselimäärällä ei ole tietoa. Testatuilla esinemäärillä kulutuksen määrän kasvaminen ei ollut kuitenkaan äärimmäinen. [10]

KD-puu on suhteellisen kustannustehokas puun läpikäymisessä muihin kiihdytysrakenteisiin verrattuna. Sen päivittäminen dynaamisessa kohtauksessa on kuitenkin tarkasteltaviin kiihdytysrakenteisiin verrattuna hintavaa. Kiihdytysrakenne voisi toimia kiihdytysrakenteista parhaiten staattisten kuvien renderöintiin.

BVH toimii paremmin dynaamisten kohtausten käsittelemisessä, kuin KD- tai CB-puut. Se pystyy käsittelemään ja mukautumaan kohtauksen muutoksiin paremmin. BVH:n uudelleenrakennus on laskennallisesti halvempaa, kuin KD- tai CB-puun. CB-puun rakennus on laskennallisesti vaativaa, sillä menetelmä yhdistää CPR- ja CSR puut, joten molemmat puut joudutaan rakentamaan. Kohtauksen muuttuessa CB-puun uudelleenrakennus on siis huomattavasti vaativampaa, kuin BVH:n, jonka uudelleenrakennus on suhteellisen yksinkertaista. Rakenteen uudelleenrakennus on siis BVH:ssä nopeampaa ja vähemmän vaativaa, kuin CB- tai KD-puun. Menetelmistä suorituskyvyn perusteella BVH suoriutuu siis parhaiten dynaamisessa kohtauksessa.

CB-puu on suunniteltu vähentämään aikavaativuutta perinteisessä säteenseurannassa. Se tehdään vähentämällä säteiden ja esineiden leikkauksien hintaa. Käsitel-

tävien leikkauksien määrän vähentäminen on johtanut ylimääräisten kustannusten vähenemiseen, jolloin suorituskyky on huomattavasti parantunut. CB-puu mahdollistaa kuvassa saman visuaalisen laadun, kuin perinteinen säteenseuranta. [10] Verrattuna BVH ja KD-puuhun, leikkauspisteiden määrä on CB-puussa on huomattavasti pienempi. CB-puu toimii tämän ansiosta tehokkaammin momimutkaisissa ja tiheissä kohtauksissa, kuin muut menetelmät. Leikkauspisteiden laskeminen on säteenseurannassa laskennallisesti vaativin osa, joten suorituskyky voisi kokonaisuudessa olla parempi CB-puulla, jos kohtaus on riittävän monimutkainen. Taulukossa 5.3 esitetään kiihdytysrakenteiden hyvät puolet ja haasteet algoritmien perusteella.

Taulukko 5.3: Kiihdytysrakenteiden vertailu

Rakenne	**Hyvät puolet**	**Haasteet**
BVH (Bounding Volume Hierarchy)	Helppo muodostaa, parantaa suorituskykyä reaaliaikaisessa ympäristössä	rakenne vaatii säännöllisiä päivityksiä, joka voi koitua laskennallisesti raskaaksi
KD-puu	Tehokas staattisissa kohtauksissa, nopea leikkauspisteen etsintä	Laskennallisesti raskas uudelleenrakennuksessa, ei optimaalinen dynaamisiin kohtauksiin
CB-puu (Compressed BVH)	Vähentää suoritettavien leikkauspisteiden määrää, sopii monimutkaisiin ympäristöihin	Monimutkainen toteuttaa, uudelleenrakennus tarpeen muuttuessa, optimoitava dynaamisille kohtauksille

Kohinanpoistoa ei voida suoraan verrata kiihdytysrakenteisiin, sillä optimointimenetelmä ei keskity säteenseurannan tuottaman kuvan käsittelyyn. Kohinanpoistoa voidaan käyttää yhdessä kiihdytysrakenteiden kanssa. Kohinanpoistossa voidaan kuitenkin vertailla kohinanpoistomenetelmiä toisiinsa. Tilasuodatuksen etuna on se, että sen käyttäminen ei tuota ajallista viivettä, mikä mahdollistaa välittömän reagoinnin muuttuviin olosuhteisiin. Tilasuodatus voi kuitenkin aiheuttaa kuvaan epätarkkuutta ja epäonnistua säilyttämään kulmia ja tekstuureja. Tilamuodostus ei ole laskennallisesti kovin raskasta, sillä jokainen pikseli käsitellään itsenäisesti vain lähellä sijaitsevien pikselien perusteella. Aikakeräys vähentää kuvissa värähtelyä ja

parantaa vakautta. Aikakeräyksen vaikutuksesta aiheutuu kuitenkin aikaviivettä. Aikakeräys käyttää ja prosessoi vanhoja kuvaruutuja, joka vaatii huomattavan määrän resursseja. Koneoppimisen käyttäminen tuottaa yleensä hyvälaatuisia kuvia, koska se pystyy säilykäyttäminenttämään yksityiskohdat ja tekstuurit kohinaa poistaessa. Koneoppimisen käyttäminen kohinanpoistossa on laskennallisesti vaativaa. Koneoppimisen voi aiheuttaa ajallista viivettä ja kuvasta voi tulla ajallisesti epävakaa. Epävakautta voidaan kuitenkin korjata ajallisen vakauttamisen metodeilla. Näiden käyttäminen kuitenkin käyttää vielä enemmän laitteen resursseja ja voi tuottaa viivettä. [21]

Kohinanpoistomenetelmistä parhaan kuvan tuottaa koneoppiminen, mutta menetelmä on laskennallisesti vaativa ja voi luoda kuvaan epävakautta. Aikakeräys tuottaa myös hyvälaatuista kuvaa, mutta voi aiheuttaa viivettä. Tilasuodatus ei ole laskennallisesti vaativaa, mutta kuvan laatu saattaa kärsiä prosessissa. Taulukossa 5.4 esitetään kohinanpoistotekniikoiden hyvät puolet ja haasteet algoritmien perusteella.

Taulukko 5.4: Kohinanpoistotekniikoiden vertailu

Tekniikka	**Hyvät puolet**	**Haasteet**
Tilasuodatus	Nopeasti reagoiva tilan muutoksiin, laskennallisesti kevyt	Kuvan värähtely, visuaaliset epätäydellisyydet, reunojen ja tekstuurien säilyttäminen haasteellista
Aikakeräys	Parantaa kuvan vakautta, vähentää kohinaa tehokkaasti	Saattaa aiheuttaa viivettä, laskennallisesti raskas
Koneoppiminen	Tuottaa korkealaatuisia kuvia, säilyttää yksityiskohdat	Laskennallisesti vaativa, mahdollinen ajallinen epävakaus, resursseja vaativa vakauttaminen

6 Yhteenveto

Tässä tutkielmassa käsiteltiin säteenseurannan optimointimenetelmiä, keskittyen erityisesti reaaliaikaisiin sovelluksiin. Tutkielmassa tarkasteltiin ensin säteenseurannan toimintaa ja fyysisen valon toimintaa. Säteenseurannan tominnassa käytiin läpi sen matemaattinen toiminta ja algoritmit. Optimointimenetelmistä käsittelyyn nousivat kiihdytysrakenteet ja kohinanpoisto. Kiihdytysrakenteista valittiin BVH, KD-puu ja CB-puu.

Säteenseuranta on graafisen renderöinnin tekniikka, jolla pyritään mallintamaan valon käyttäytymistä mahdollisimman todenmukaisesti. Säteenseurannan optimointia tarvitaan, koska se on laskennallisesti vaativa prosessi, erityisesti reaaliaikasessa ympäristössä. Keskeisiä optimointimenetelmiä säteenseurannassa ovat kiihdytysrakenteet, joista tutkielmaan valittiin BVH, KD-puu ja CB-puu. Menetelmien tarkoitus on parantaa suorituskykyä vähentämällä laskentatarvetta. Toinen keskeinen optimointimenetelmä on kohinanpoisto, jonka tarkoitus on parantaa kuvan laatua poistamalla kohinaa.

Kiihdytysrakenteet rajoittavat säteenseurannan laskentaa, mikä vähentää suoritettavien laskutoimitusten määrää. Kiihdytysrakenteista parhaiten suoriutuva reaaliaikaisessa ympäristössä oli BVH. Kiihdytysrakenteet joudutaan rakentamaan uudelleen jokaisen kohtauksen muutoksen jälkeen. BVH on helppo muodostaa ja se on laskennallisesti edullisempaa, kuin KD- ja CB-puiden muodostaminen. CB-puu pystyy vähentämään suoritettavien leikkauspisteiden määrää, joka voisi mahdollistaa

tehokkaamman toiminnan monimutkaisessa ympäristössä. Leikkauspisteiden määrän väheneminen voi mahdollistaa BVH:ta tehokkaamman suorituskyvyn tietyissä ympäristöissä. CB-puu voisi olla siis parempi valinta ympäristöissä, joissa on useita yksityiskohtia ja paljon esineitä. KD-puun todettiin olevan kallis uudelleenrakentaa, joten se ei ole optimaalinen kiihdytysrakenne dynaamisiin kohtauksiin. KD-puu mahdollistaa kuitenkin nopean ja kustannustehokkaan leikkauspisteen etsinnän, joten staattisissa kohtauksissa KD-puun käyttäminen voisi olla paras valinta.

Kohinanpoistossa koneoppimisen todettiin tuottavan parhaan laatuisen kuvan. Koneoppiminen voi kuitenkin aiheuttaa ajallista epävakautta, joka voi näkyä dynaamisessa kohtauksessa. Koneoppimisen käyttäminen ja epävakauden korjaaminen on laskennallisesti vaativaa. Aikakeräys hyödyntää edellisistä ruuduista tietoa, jonka avulla saadaan tuotettua hyvälaatuista kuvaa kohinaa poistaessa. Aikakeräys vähentää myös kuvan värähtelyä ja säilyttää kuvien välillä visuaalisen jatkuvuuden, tehden metodista hyödyllisen dynaamisessa kohtauksessa. Aikakeräys voi kuitenkin aiheuttaa viivettä, joka voi aiheuttaa ongelmia reaaliaikaisissa sovelluksissa. Tilasuodatus on nopea ja yksinkertainen suorittaa, mutta voi aiheuttaa metodeista huonoimman kuvan visuaalisen laadun.

Tutkielman tutkimusta ja päätelmiä rajoitti muutama asia. Vertailussa ei tarkasteltu CB-puun suorituskykyä suuremmissa ja monimutkaisemmissa kohtauksissa. CB-puun todettiin parantavan suorituskykyä tällaisessa ympäristössä. Tästä ei kuitenkaan ollut varmuutta testidatan puutteen takia.

Jatkotutkimusta aiheesta voitaisiin tehdä esimerkiksi tarkastelemalla CB-puun suorituskykyä suuremmissa kohtauksissa. Kohinanpoistoa koneoppimisen avulla voitaisiin myös tutkia. Jos koneoppimisen tuottamia epävakauksia saataisiin korjattua tai vähennettyä, metodista voisi kehittyä tehokas kohinanpoistomenetelmä.

Lähdeluettelo

[1] Y. Deng, Y. Ni, Z. Li, S. Mu ja W. Zhang, "Toward Real-Time Ray Tracing: A Survey on Hardware Acceleration and Microarchitecture Techniques", *ACM Comput. Surv.*, vol. 50, nro 4, s. 1–7, elokuu 2017, ISSN: 0360-0300. DOI: 10.1145/3104067. url: https://doi-org.ezproxy.utu.fi/10.1145/3104067.

[2] E. Vasiou, K. Shkurko, I. Mallett, E. Brunvand ja C. Yuksel, "A detailed study of ray tracing performance: render time and energy cost", *The Visual Computer*, vol. 34, s. 875–883, kesäkuu 2018. DOI: 10.1007/s00371-018-1532-8.

[3] NVIDIA Corporation, *Ray Tracing*, https://developer.nvidia.com/discover/ray-tracing, Viitattu: 07-04-2024.

[4] Henrik, *Ray tracing diagram*, Image under CC BY-SA 4.0 license, Teksti käännetty suomeksi Kasperi Juutisen toimesta, 2008. url: https://upload.wikimedia.org/wikipedia/commons/8/83/Ray_trace_diagram.svg.

[5] S. Park ja N. Baek, "A Shader-Based Ray Tracing Engine", *Applied Sciences*, vol. 11, nro 7, 2021, ISSN: 2076-3417. DOI: 10.3390/app11073264. url: https://www.mdpi.com/2076-3417/11/7/3264.

[6] E. Haines ja T. Akenine-Möller, toim., *Ray Tracing Gems*. Apress, 2019, http://raytracinggems.com.

[7] T. Whitted, "An improved illumination model for shaded display", *Commun. ACM*, vol. 23, nro 6, s. 343–349, kesäkuu 1980, ISSN: 0001-0782. DOI: 10.1145/358876.358882.

[8] D. J. Eck, *Introduction to Computer Graphics*, Image under Creative Commons Attribution-Noncommercial-ShareAlike 4.0 License. Tekstiä käännetty ja lisätty kuvaan Kasperi Juutisen toimesta. url: https://math.hws.edu/graphicsbook/c8/ray-tracing-2d.png.

[9] A. B. Samardžić, D. Starčević ja M. Tuba, "An Implementation of Ray Tracing Algorithm for the Multiprocessor Machines", *Yugoslav Journal of Operations Research*, vol. 16, nro 1, s. 125–135, 2006.

[10] K. Ahmed, S. A. Sony, T. Tanjimat, M. M. Ahmed Shibly, M. Rahman ja T. Jabid, "Cone Based Tree to Improve Ray Tracing", teoksessa *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 2021, s. 709–714. DOI: 10.1109/ICCCIS51004.2021.9397073.

[11] National Aeronautics and Space Administration, Science Mission Directorate, *Wave Behaviors*, Viitattu: 19-05-2024, 2010. url: https://science.nasa.gov/ems/03_behaviors.

[12] A. Peuhkurinen. ja T. Mikkonen., "Real-time Human Eye Resolution Ray Tracing in Mixed Reality", teoksessa *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - GRAPP*, INSTICC, SciTePress, 2021, s. 169–176, ISBN: 978-989-758-488-6. DOI: 10.5220/0010205701690176.

[13] S. Mostafawy, O. Kermani ja H. Lubatschowski, "Virtual Eye: retinal image visualization of the human eye", *IEEE Computer Graphics and Applications*, vol. 17, nro 1, s. 8–12, 1997. DOI: 10.1109/38.576849.

[14] I. Csoba ja R. Kunkli, ”Rendering algorithms for aberrated human vision simulation”, English, *Visual Computing for Industry Biomedicine, and Art*, vol. 6, nro 1, s. 5, joulukuu 2023, Copyright - © The Author(s) 2023. This work is published under http://creativecommons.org/licenses/by/4.0/ (the “License”). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2023-12-05. url: https://www.proquest.com/scholarly-journals/rendering-algorithms-aberrated-human-vision/docview/2890359776/se-2.

[15] Y. Kim, ”REAL-TIME RAY TRACING REFLECTIONS AND SHADOWS IMPLEMENTATION USING DIRECTX RAYTRACING”. url: https://api.semanticscholar.org/CorpusID:264352210.

[16] D. Kopta, K. Shkurko, J. Spjut, E. Brunvand ja A. Davis, ”Memory Considerations for Low Energy Ray Tracing”, *Computer Graphics Forum*, vol. 34, nro 1, s. 47–59, 2015. DOI: https://doi.org/10.1111/cgf.12458.

[17] B. Smits, ”Efficiency issues for ray tracing”, teoksessa *ACM SIGGRAPH 2005 Courses*, sarja SIGGRAPH ’05, Los Angeles, California: Association for Computing Machinery, 2005, 6–es, ISBN: 9781450378338. DOI: 10.1145/1198555.1198745. url: https://doi-org.ezproxy.utu.fi/10.1145/1198555.1198745.

[18] J. Hendrich, A. Pospíšil, D. Meister ja J. Bittner, ”Ray Classification for Accelerated BVH Traversal”, *Computer Graphics Forum*, vol. 38, nro 4, s. 49–56, 2019. DOI: https://doi-org.ezproxy.utu.fi/10.1111/cgf.13769. eprint: https://onlinelibrary-wiley-com.ezproxy.utu.fi/doi/pdf/10.1111/cgf.13769. url: https://onlinelibrary-wiley-com.ezproxy.utu.fi/doi/abs/10.1111/cgf.13769.

[19] J. L. Bentley, "Multidimensional binary search trees used for associative searching", *Commun. ACM*, vol. 18, nro 9, s. 509–517, syyskuu 1975, ISSN: 0001-0782. DOI: 10.1145/361002.361007. url: https://doi-org.ezproxy.utu.fi/10.1145/361002.361007.

[20] L. Fan, F. Zhang, H. Fan ja C. Zhang, "Brief review of image denoising techniques", English, *Visual Computing for Industry Biomedicine, and Art*, vol. 2, nro 1, s. 1–12, heinäkuu 2019, Copyright - Visual Computing for Industry, Biomedicine, and Art is a copyright of Springer, (2019). All Rights Reserved.; © 2019. This work is published under http://creativecommons.org/licenses/by/4.0/. Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2023-12-06. url: https://www.proquest.com/scholarly-journals/brief-review-image-denoising-techniques/docview/2264220387/se-2.

[21] J. Kim, *What is Denoising?*, https://blogs.nvidia.com/blog/what-is-denoising/, Viitattu: 04-05-2024, 2023.

[22] V. Havran, "Heuristic ray shooting algorithms", 2000. url: https://api.semanticscholar.org/CorpusID:12095296.

[23] J. Bikker, "Real-time Ray Tracing through the Eyes of a Game Developer", teoksessa *2007 IEEE Symposium on Interactive Ray Tracing*, 2007, s. 1–10. DOI: 10.1109/RT.2007.4342584.

[24] L. O. Simonsen, N. Thrane ja P. Ørbæk, "A comparison of acceleration structures for GPU assisted ray tracing", *Master's thesis, University of Aarhus*, s. 31–86, 2005.