Joni Kamunen

Nelipuu ja sen käyttö videonpakkauksessa

Tietotekniikan kandidaatintutkielma

14. maaliskuuta 2022

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Joni Kamunen

Yhteystiedot: joni.m.kamunen@student.jyu.fi

Työn nimi: Nelipuu ja sen käyttö videonpakkauksessa

Title in English: Quadtree and its usage in video encoding

Työ: Kandidaatintutkielma

Sivumäärä: 28+0

Tiivistelmä: Tämä kandidaatin tutkielma on toteutettu kirjallisuuskatsauksena. Tutkielman tarkoituksena on tutkia, miten nelipuuta käytetään videonpakkauksessa. Tässä tutkielmassa tutustutaan tiedonpakkaukseen ja tilansäästöön neli- ja kasipuu-tietorakenteissa, sekä miten nelipuita käytetään HEVC videonpakkauksessa. Tutkielmassa jaotellaan tiedonpakkaus kahteen eri kategoriaan: häviölliseen- ja häviöttömään tiedonpakkaukseen, määritellään nelipuu ja sen laajennoksia ja variantteja kuten kasipuu, R -puu ja kd -puu, sekä tutustutaan näiden puurakenteiden sovelluskohteisiin. Tutkielmassa havaitaan, että HEVC videonpakkauksessa käytetään nelipuita mm. pakattavan videokuvan ennustemallin luonnissa.

Avainsanat: nelipuu, kasipuu, tiedonpakkaus, videonpakkaus, HEVC

Abstract: This bachelor's thesis was carried out as a literature review. The goal of the thesis is to examine how quadtrees are used in video compression. This thesis gets familiar with datacompression and saving of space with quadtree and octree data structures and how quadtrees are used in HEVC video compression. Data compression is categorized into two categories: lossy and lossless data compression, followed by quadtree and its expansions and variants such as octree, R -tree and kd -tree definitions and examples of their usage in different applications. In this thesis, it is observed that HEVC video compression uses quadtrees for example to create prediction for observable frame.

Keywords: quadtree, octree, compression, videocompression, HEVC

Kuviot

Sisältö

1 Johdanto

Hierarkkiset tietorakenteet ovat laajalti käytössä ja niihin törmää erilaisissa sovelluskohteissa. Tässä tutkielmassa perehdytään tarkemmin nelipuuhun ja sen käyttöön videonpakkauksessa. Nelipuu on puumainen hierarkkinen tietorakenne. Nelipuuta ja sen laajennosta kasipuuta käytetään 2D (kuva) ja 3D (tila) informaation käsittelyssä. Nelipuut liittyvät myös vahvasti tiedon tiivistämiseen ja niitä käytetään hyödyksi sekä häviöttömässä, että häviöllisessä tiedon tiivistämisessä. Nelipuut saattavat syötetystä tiedosta riippuen itsessään jo tiivistää tietoa verrattuna tiedon listapohjaiseen esitykseen (Samet 1984, s. 218). Nelipuun suurimmat vahvuudet liittyvät niiden tarjoamiin algoritmeihin, jotka mahdollistavat mm. tiedon hakemisen nelipuusta tehokkaasti (Sonka, Hlavacc ja Boyle 1999, s. 51). Tutkielmassa tutustutaan myös nelipuun k-d- ja R -puu variantteihin.

Videokuva on sarja kuvia, jotka näytetään peräjälkeen katsojalle. Videokuvaa pakataan, jotta voidaan mahdollistaa nykyaikaisen videokuvan lähetys tai tallennus mahdollisimaan nopeasti ja siten, että se vie mahdollisimman vähän tilaa. Tätä varten on kehitetty videonpakkausmenetelmiä, joiden avulla raakavideo muutetaan (usein) häviöllisesti pienempään muotoon ja uudelleen videota katsellessa muutetaan pakatusta muodosta takaisin katsottavaan muotoon. Tässä tutkielmassa tullaan avaamaan videonpakkausta tarkemmin HEVC videonpakkausmenetelmän näkökulmasta.

Tutkielma on toteutettu kirjallisuuskatsauksena. Rakenteellisesti tutkielma etenee tiedonpakkaukseen määrittelystä neli- ja kasipuu -tietorakenteisiin, sekä niiden variantteihin ja sovelluskohteisiin. Sovelluskohteista erityisesti käsitellään nelipuiden käyttöä videonpakkauksessa. Aivan tutkielman lopussa on yhteenveto.

2 Tiedonpakkaus

Tiedonpakkaus on tärkeässä osassa puurakenteita. Tiedonpakkauksella pyritään mm. säästämään käytettävää tallennustilaa. Seuraavassa kappaleessa käydään läpi mitä tarkoittaa tiedonpakkaus ja mitä tarkoittavat häviötön- ja häviöllinen tiedonpakkaus. Lisäksi käydään läpi Huffman koodauksen perussäännöt.

2.1 Tiedonpakkaus

Tiedonpakkausalgoritmi ottaa syötteen ja muodostaa siitä pakatun version, jonka koko on pienempi kuin alkuperäisen syötteen. Pakattu tieto puretaan käyttämällä tiedonpurkualgoritmia, joka ottaa syötteenä pakatun tiedon ja tuottaa siitä uudelleenmallinnuksen. Pakkaustekniikat voidaan jakaa kahteen eri kategoriaan: häviöttömiin- ja häviöllisiin pakkausmenetelmiin (Khalid Sayood 2017, s. 4). Bing (2015) määrittelee kompressoinnin videonpakkauksesta näin: "Pakkaus vähentää videon tallennuksen ja tiedonsiirron hintaa muuttamalla datan vaatimaan vähemmän tilaa". (Bing 2015, s. 29, suomennos minun).

2.1.1 Huffman koodaus

Huffman koodauksen kehitti David Huffman (Huffman 1952, s. 1098). Huffman koodauksessa ideana on muuttaa koodattavaksi annettu viesti tai teksti mahdollisimman vähän tilaa vieväksi siten, että annetaan mahdollisimman lyhyet symbolit yleisimmille kombinaatioille (Huffman 1952, s. 1098).

Määritellään, että seuraavissa Huffman koodin rajoitteita käsittelevissä listauksissa koodausnumerolla tarkoitetaan Huffman koodin sisäistä yksittäistä alkioita, esimerkiksi koodin "r195" numeroa "1". Huffman koodille (Huffman 1952, s. 1098) asetetaan seuraavat rajoitteet:

1. "Kaksi koodia ei koostu identtisistä koodausnumeroiden järjestelyistä."
   (Huffman 1952, s. 1098, suomennos minun)
2. "Koodi pitää toteuttaa siten, että ei tarvita ulkoista indikaattoria erittelemään missä edellinen koodi loppuu ja seuraava alkaa sen jälkeen kun koodin aloituspiste on tie-

dossa."
(Huffman 1952, s. 1098, suomennos minun)

Huffman jatkaa, että optimaaliselle Huffman koodille on lisäksi kolme lisärajoitusta (Huffman 1952, s. 1099).

3. $L(1) < L(2) < ... < L(N-1) = L(N)$, millä tarkoitetaan, että todennäköisemmällä koodilla on lyhyempi tai yhtä pitkä koodi kuin seuraavaksi todennäköisimmällä. (Huffman 1952, s. 1099)
4. Minimissään kahdella, mutta maksimissaan yksittäisten alkioiden symbolien maksimimäärällä (esimerkiksi eri koodityyppejä voivat olla 1, 2, 3, joka tarkoittaa että symboleita on 3 erilaista) eri koodeja, voi sisältää vain viimeisimmiltä alkioilta eroavan muuten identtisen koodin keskenään.
(Huffman 1952, s. 1099)
5. Koodin $L(N-1)$ (Koodi, jonka todennäköisyys on toistaiseksi toiseksi pienin) kaikki edeltävät järjestykset tai etuliitteet täytyy olla käytettynä Huffman koodissa.
(Huffman 1952, s. 1099)

Khalid Sayoodin (2017, s. 72, 74, 75) mukaan Huffman koodausta käytetään esimerkiksi häviömättömän kuvan-, tekstin- ja äänenpakkauksen sovelluskohteissa. Huffman koodaus on häviötöntä tiedonpakkausta, johon perehdymme seuraavassa kappaleessa.

2.2 Häviötön tiedonpakkaus

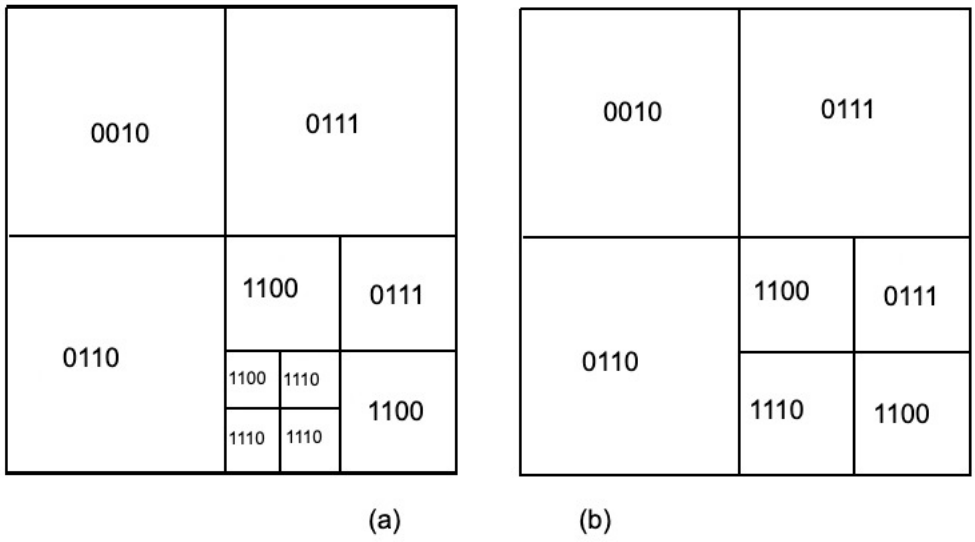
Häviötön pakkaus ei kadota tietoa pakkauksessa. Pakkaus toteutetaan esittämällä pakattu tiedosto käyttäen vähemmän tavuja.Khalid Sayoodin (2017, s. 4) mukaan häviöttömässä pakkausmenetelmässä alkuperäinen tieto, joka pakataan on täsmälleen sama kuin samasta pakkauksesta purettu tieto. Häviöttömän tiedonpakkauksen ehdoton etu häviölliseen on, että voidaan olla varmoja ettei tietoa katoa tai muutu pakkauksessa.

Nelipuu on toteutettu alunperin vaihtoehdoksi binääriselle listalle tarkoituksenaan säästää tilaa yhdistämällä saman arvon sisältäviä alueita yhteen (Samet 1985, s. 973). Tulemme palaamaan aiheeseen vielä myöhemmin tutkielmassa, mutta esimerkiksi kuvatiedolle nelipuu

itsessään voi säästää häviöttömästi tilaa verrattuna siihen, että annettu kuvatieto esitetään listaesityksenä. Kuitenkin jos kuvan pikseleiden arvot eroavat viereisten pikseleiden arvosta (vrt. shakkilauta -kuvio), ei nelipuista saada tilanpakkaushyötyä irti vaan listaesitys saattaa viedä jopa vähemmän tilaa (Samet 1984, s. 218). Esimerkkiä jatkaen voitaisiin häviöttömästi tallentaa kuvan väriarvot niiden esiintymistiheyden mukaan käyttäen esimerkiksi Huffman koodausta 2.1.1.

2.3 Häviöllinen tiedonpakkaus

Häviöllisiin pakkaustekniikoihin kuuluu osittainen tiedon häviäminen tietoa pakatessa. Tietoa joka on pakattu käyttämällä häviöllistä tekniikkaa ei pysty palauttamaan tai rakentamaan täysin samanlaisena uudestaan (K. Sayood 2006, s.5). Khalid Sayoodin (2006) mukaan häviöllisessä pakkamisessa on hyötynä se, että jos voidaan hyväksyä tiedon epätarkkuuden kasvaminen (esimerkiksi kuvan sumentuminen), saadaan aikaan paljon parempi pakkauskerroin verrattuna häviöttömiin pakkausmenetelmiin. Esimerkiksi HEVC videonpakkaus on häviöllinen pakkausmenetelmä, vaikkakin HEVC:iin on toteutettu myös häviöttömän pakkauksen tuki (Sullivan ym. 2012, s. 1664).



Kuvio 1. Esimerkki häviöllisestä pakkauksesta nelipuu-tietorakenteessa. (a) osiossa pakattava kuva ja (b) pakkauksen jälkeen purettu kuva. Pakkauksessa on tiivistetty sulauttamalla 11 -alkuiset alipuut.

3 Puurakenteet spatiaalisen tiedon esittämisessä

Nelipuita käytetään laajasti esimerkiksi erilaisen alueellisen tiedon esittäjänä. Nelipuut ovat vaihtoehtoina sovelluksilla kun käsitellään mm. kuvia, videota tai vaikka paikkatietoa. Tässä kappaleessa määritellään nelipuu, nelipuun perusoperaatiot ja sen yleisimmät laajennokset. Nelipuun käytössä esimerkiksi kuvadatalle voidaan saavuttaa sekä hyvä pakkaussuhde, että tehokkaat nelipuuta hyödyntävät algoritmit.

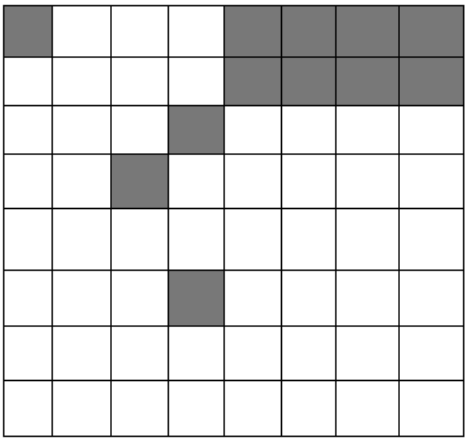
3.1 Nelipuun määritelmä

Nelipuu määritellään Hunterin ja Steiglitzin (1979, s. 123, suomennos minun) mukaan seuraavasti: "Nelipuu on puurakenne, jonka solmut ovat joko lehtisolmuja tai niillä on neljä lapsisolmua. Lapset ovat järjestetty 1, 2, 3 ja 4.". Shihin (2010, s. 223) mukaan puurakenne on nelipuu, koska sen kaikilla ei lehtisolmuilla on neljä lapsisolmua. Määritellään siis nelipuu puurakenteeksi, jonka solmuilla on aina joko ei yhtään tai neljä lasta.

Nelipuun hyöty kuvien esittämisessä on yksinkertaisten algoritmien olemassaolo kuvien lisäykselle, alueiden laskemiselle ja hetkellisille tilastojen laskemiselle (Sonka, Hlavacc ja Boyle 1999, s. 51). Sonka, Hlavacc ja Boyle 1999, (s. 51) mukaan nelipuiden suurin heikkous on niiden riippuvaisuus paikkaan, orientatioon ja objektien relatiiviseen kokoon. Tästä esimerkkinä Sonka, Hlavacc ja Boyle 1999, (s. 51) mainitsee, että kaksi hyvinkin samannäköistä kuvaa hyvin pienillä eroavaisuuksilla tai pieni kuvan siirtyminen johonkin suuntaan voi tuottaa kaksi hyvinkin erilaista nelipuuta.

3.1.1 Aluepohjainen neli- ja kasipuu

Nelipuun määritelmän mukaan aluepohjaisen nelipuun solmut ovat joko lehtisolmuja tai sitten viittaavat neljään lapsisolmuun. Jokaista näistä solmuista vastaa osa-alue. Lehtisolmun sisältämä osa-alue voidaan esittää yhdellä data-alkiolla. Aluepohjaisella nelipuulla voidaan kuvata kuvatietoa, jossa lehtisolmulla voisi olla esimerkiksi kaksi eri tietokenttää: sijainti kuvassa ja pikselin väri (Samet 1984, s. 189).



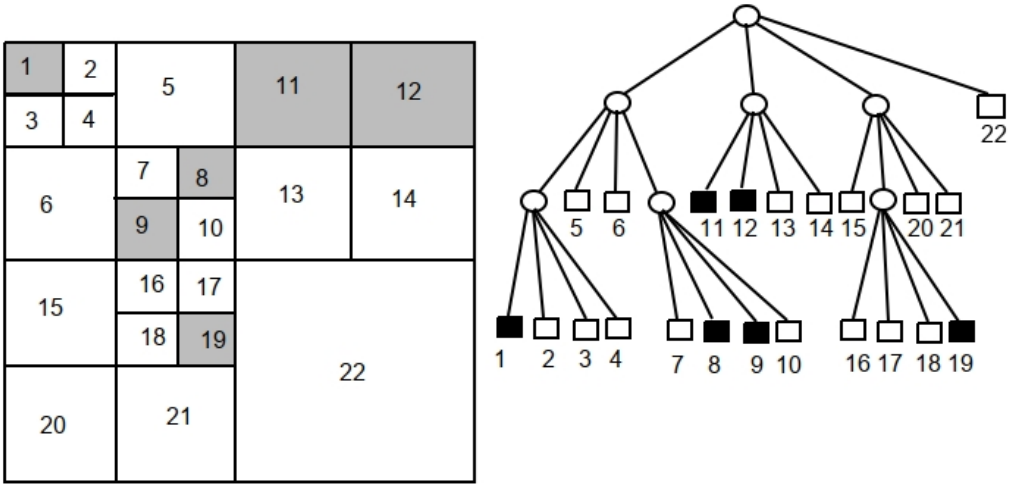
Kuvio 2. Aluepohjaiselle nelipuulle annettava 2D kuvan pikselitieto

Esimerkiksi kuviossa 2, jossa kuva on jaettu valmiiksi pikselin kokoisiin alueisiin jaetaan 4 samankokoiseen alueeseen pohjois-etelä ja länsi-itä suunnassa, näitä alueita kutsutaan kvadranteiksi. Ensimmäisen kvadranteiksi jaon jälkeen, jos tutkittavan kvadrantin sisältämät pikselit sisältävät toisistaan eroavia väriarvoja, niin kyseinen kvadrantti jaetaan uudestaan neljään osaan. Tätä toistetaan, kunnes ollaan määritellyllä maksimisyvyydellä nelipuussa, tai tutkittavassa kvadrantissa on vain samanarvoisia pikseleitä. Tällöin jäljelle jäävästä osasta tulee lehtisolmu (Samet 1984, s. 189). Lopputulos on kuviossa 3.

Kasipuu jako tehdään vastaavasti kuin nelipuussa, mutta esimerkiksi aluepohjaisessa kasipuussa (vrt. aluepohjainen nelipuuhun) alue on kuutio ja se jaetaan kahdeksaan pienempään samankokoiseen kuutioon. Puunäkymässä kasipuu eroaa nelipuusta vain siten, että sillä on neljän lapsisolmun sijasta niitä kahdeksan (Samet ja Webber 1988, s. 54).

3.1.2 Pistepohjainen neli- ja kasipuu

Pistepohjaisessa nelipuussa jokainen datapiste on puun solmu. Jokaisella näistä solmuista on neljä lapsisolmua, joita kutsutaan kvadranteiksi. Lapsisolmut ovat järjestyksessä: Vasen ylä- (NW), oikea ylä- (NE), vasen ala- (SW), oikea alasolmu (SE). Jokainen datapiste on uniikki. Pistepohjaisessa nelipuussa alueiden jakautuminen ja nelipuun tasapainoisuus riippuu hyvin paljon siitä , missä järjestyksessä data nelipuuhun syötetään (katso kuvio 4) (Samet 1984, s. 189). Esimerkiksi tilanteessa, jossa tiedot syötetään siten, että datapisteet menevät lineaari-



Kuvio 3. Esimerkki aluepohjaisesta nelipuusta. Kuvio on muokattu Sametin luomasta visualisaatiosta (Samet 1984, s. 191)

sesti alueen yhdestä kulmasta vastakkaiseen kulmaan 5, on lopputuloksena syntynyt linkitetty lista, jolloin ei saada käyttöön puurakenteen hyötyjä. (Samet 1984, s. 230).

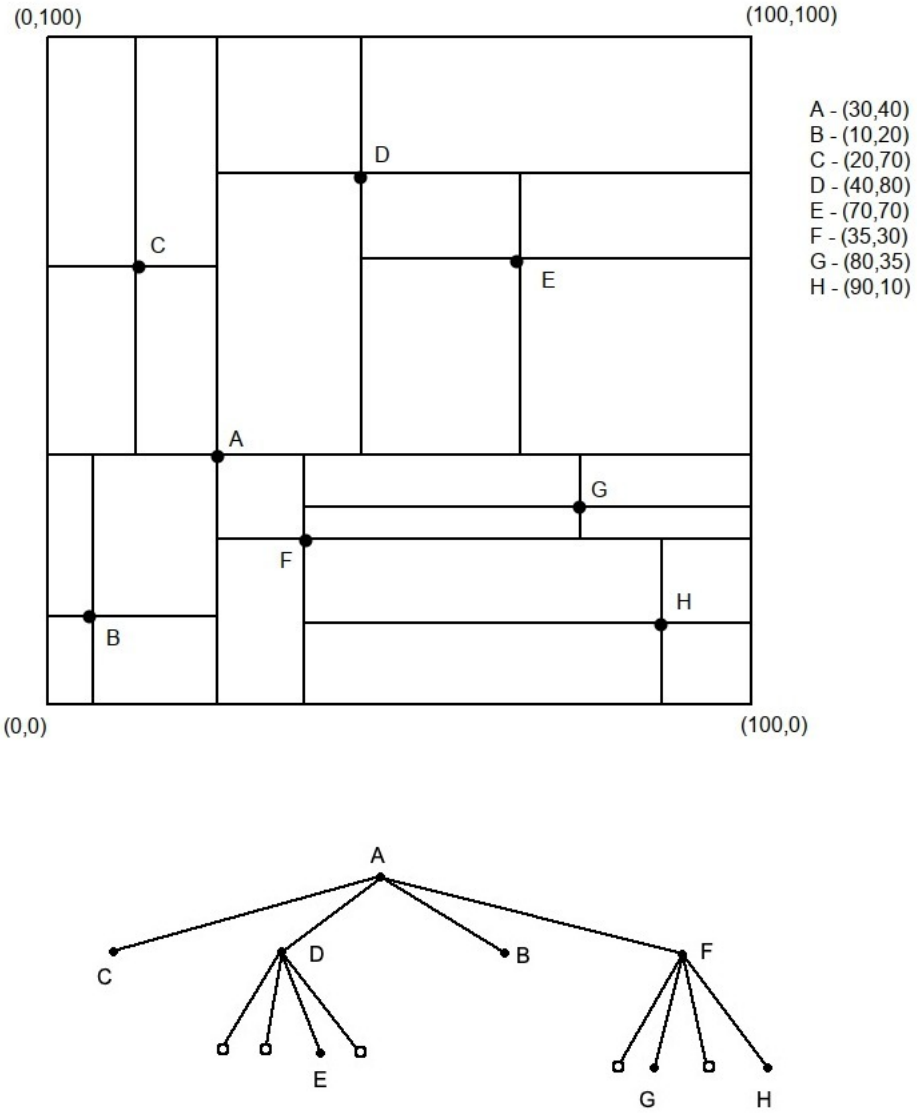
Pistepohjaisen nelipuun jossa on M kappaletta arvoja maksimisyvyys on $(M-1)$. Tämänlaisessa tilanteessa yksi arvo on syötetty puun jokaiselle tasolle (kts. kuvio 5). Pistepohjaisen nelipuun minimisyvyys on Sametin mukaan $log_4(3*m)$, tässä tapauksessa puun jokainen taso olisi täytetty. (Samet 1984, s. 236)

Vastaavasti kuin aluepohjaisessa variantissa pistepohjainen kasipuu on vastaava kuin pistepohjainen nelipuu, mutta sen sijaan, että alue jaettaisiin kvadrantteihin, alue jaetaankin kahdeksaan osaan eli oktantteihin (Samet 1984, s. 227).

3.1.3 Tiedon haku

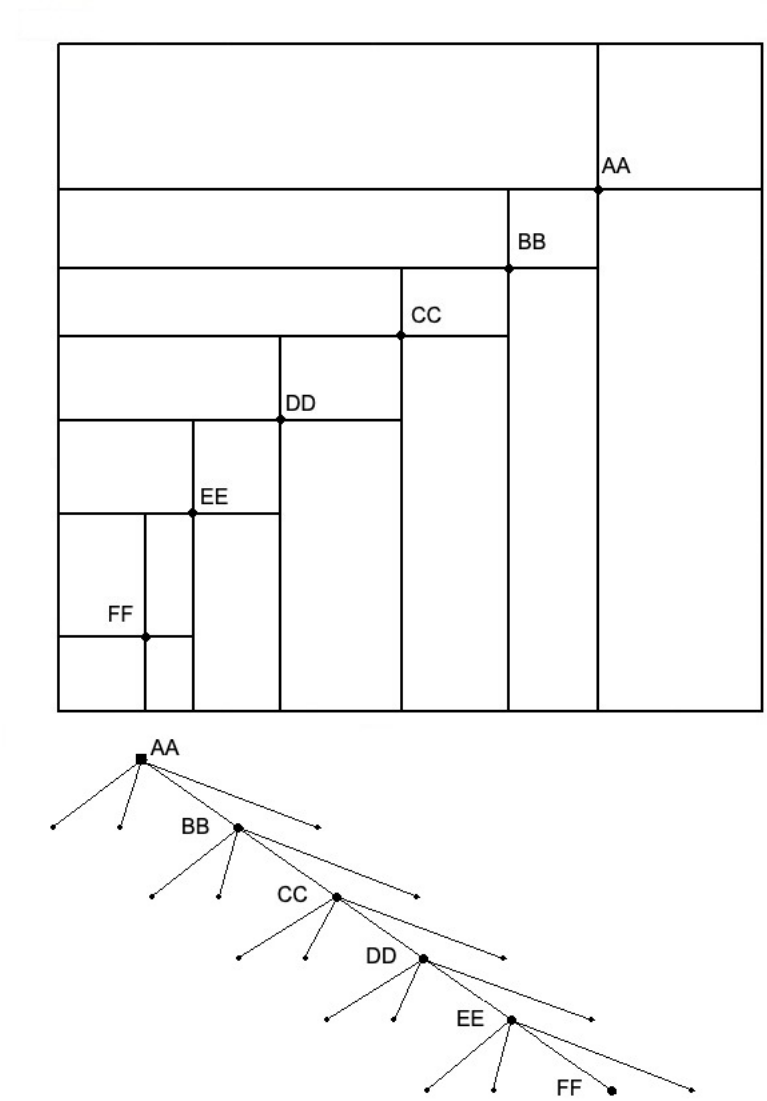
(Samet ja Webber 1988) esittelee solun tietojen hakua rasterimuotoisesta datasta aluepohjaista neli- tai kasipuuta hyödyntäen käyttäen esimerkkinä tietyn pikselin väritiedon hakemista kuvasta tai kolmiulotteisesta datasta.

(Samet ja Webber 1988) mukaan etsittäessä pistettä aluepohjaisesta nelipuusta se vaatii ne-



Kuvio 4. Esimerkki pistepohjaisesta nelipuusta. Tähän nelipuuhun data on syötetty järjestyksessä A, B, C, D, E, F, G, H. Kuvio on muokattu Sametin esittämän visualisaation pohjalta (Samet 1984, s.231).

lipuun rakenteen tutkimista. Etsintä aloitetaan puun juurisolmusta käyttämällä alueen keskipisteen X ja Y koordinaatteja. Tällöin vertaamalla etsittävän pisteen koordinaatteja juurisolmun koordinaatteihin saadaan selville, että mihin nelipuun neljännekseen etsittävä solmu kuuluu. Esimerkiksi jos etsittävän solmun X ja Y koordinaatti on pienempi kuin tarkastel-



Kuvio 5. Esimerkki pistepohjaisesta nelipuusta, johon data on syötetty vastaavasti AA, BB, CC, DD, EE, FF, GG, HH. Tässä tapauksessa data on kuitenkin syötetty pahimmassa mahdollisessa järjestyksessä, jolloin nelipuut hyödyt jäävät vähäisiksi. Kuvio on muokattu Sametin esittämän visualisaation pohjalta (Samet 1984, s.231).

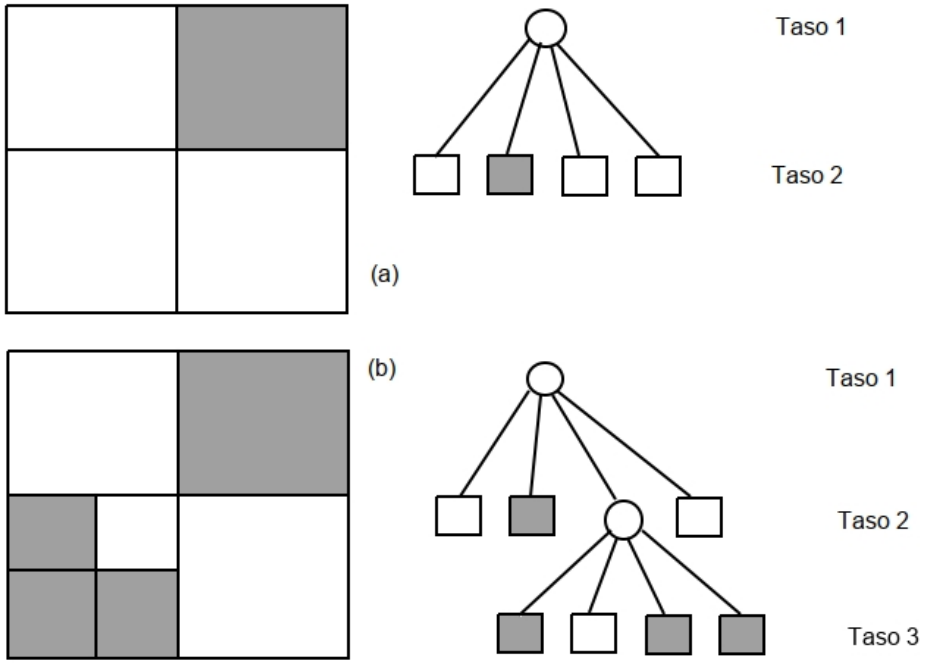
tavan solmun *X* ja *Y* koordinaatit, tällöin etsittävä solmu löytyy alueen alavasemmalla *SW* suunnassa sijaitsevasta kvadrantista. Tätä algoritmia toteutetaan rekursiivisesti, kunnes kyseinen kvadrantti sisältää vain etsittävän pisteen, eikä seuraavaa nelipuun solmua. (Samet ja

Webber 1988, s. 58).

Lehtisolmun etsintään menevä suoritusaika on verrannollinen siihen, miten syvälle nelipuuta joudutaan kulkemaan, ennenkuin oikea lehtisolmu löytyy (Samet ja Webber 1988, s. 58). Pistepohjaisella nelipuulla tämä tarkoittaa pahimmillaan $O(n)$ suoritusaikaa, missä n on alkioiden määrä, mikäli puu on on mahdollisimman epätasapainossa, kuten kuvassa 5. Sametin mukaan pistepohjaisessa nelipuussa joka olisi täysin tasapainossa ja kaikki lehtisolmut täytetty syvyys olisi $log_4(3n)$, missä n tarkoittaa datapisteiden määrää (Samet 1984, s. 236).

3.1.4 Tiedon muutokset nelipuussa

Kuvasovelluksissa lisäyksia ja poistoja ei yleensä juuri tarvita. Tarpeen aiheuttaa kuitenkin se, että jos kuvaa editoidaan niin pikseliarvon muutos muuttaa myös koko nelipuun rakennetta. Tästä kärjistettynä esimerkkinä kuvio 6, jossa osaa (a) muokataan lisäämällä mustia kohtia pienellä alueelle kuvion vasempaan alalaitaan (b).



Kuvio 6. Kuvan muokkauksen vaikutus vastaavaan nelipuuhun

3.1.5 Lisäys nelipuuhun

Samet 1984 käyttää tekstissään MX (matrix) nelipuuta kuvaamaan pistepohjaista dataa aluepohjaisessa nelipuussa. Erona MX nelipuulla ja perinteisin aluepohjaisella nelipuulla on, että MX nelipuussa lehtisolmuilla on vain kaksi eri arvoa; musta ja tyhjä (valkoinen) (Samet 1984, s. 233).

Täytettäessä MX nelipuuta lisätään uudet datapisteet etsimällä niitä. Etsintä perustuu esimerkiksi etsittävän tiedon x ja y koordinaatteihin (Samet 1984, s. 233) (katso 3.1.3). Lisäyksessä haku päättyy epäonnistuneena lehtisolmuun. haku on epäonnistunut jos lehtisolmun arvo on NIL, eli lehtisolmulla ei ole vielä arvoa. Tällöin lehtisolmu jaetaan neljään osaan, kunnes tarkasteltavan lehtisolmun alue on yksittäinen piste/alue. Tämä lopputuloksena löydetty alue asetetaan etsittäväksi arvoksi. (Samet 1984, s. 233).

Pistepohjaisissa nelipuissa pisteiden lisäykset ja poistot ovat sovelluksesta riippuen yleisempiä. Pisteen lisääminen nelipuuhun tapahtuu etsimällä (kts. 3.1.5) lisättävää arvoa olemassaolevasta nelipuusta. Jokaisessa solmussa tehdään vertailu (kts. 3.1.3) ja kun lopulta päästään puun loppuun löytämättä lisättävää tietoa, lisätään se sinne omalle paikalleen (Samet 1984, s. 230).

3.1.6 Poisto nelipuusta

Samet 1980 kirjoittaa, että nelipuusta poistaminen ei ole aivan yhtä helppoa kuin esimerkiksi binääripuusta. Nelipuun tapauksessa solmun poistamisen ongelma on ratkaistu sijoittamalla kaikki poistettavan solmun alla olevat lapsisolmut ja lehtisolmut uudestaan nelipuuhun (Samet 1980, s. 704). Samet 1980 mukaan uudelleen lisäämisen prosessi on usein kallis prosessoriajaltaan.

3.2 Nelipuun tallennustilan tarve

Shihin (2010, s. 223) mukaan nelipuun tallennustilan tarve on verrannollinen solmujensa määrään. Jokaisella nelipuun solmulla on osoitin kohdistettuna lapsisolmuihinsa, sekä mahdolliseen datan sijaintiin. Shih jatkaa, että Nelipuu ei ole datatoisteinen, mutta se on muu-

toksille herkkä, mistä johtuen pienikin muutos kohdedatassa voi aiheuttaa ison muutoksen sitä kuvastavassa nelipuussa, joka tekee kuvien vertailemisen puun perusteella haastavaksi. Eli kaksi kuvaa, jotka ovat melkein samannäköisiä, voivat tuottaa täysin erilaisen ja siten erikokoisen puurakenteen.

(Samet ja Webber 1988, s. 55) mukaan riippuen siitä, miten nelipuu on toteutettu, ei nelipuulla pystytä välttämättä säästämään tallennustilaa. Samet mukaan, joissain tapauksissa lista binäärilukuja voi olla ekonomisempi kuin nelipuu. Samet kuitenkin tähdentää, että nelipuun tuomat algoritmit ovat tärkeämpiä. (Samet ja Webber 1988, s. 56) nostaa esiin, että erityisesti suurimmalla osalla algoritmeista, joita tehdään nelipuu -mallinnokselle kuvasta, suoritusaika on vain murto osa verrattuna kuvan lista -mallinnokselle tehtävistä algoritmeista. Tämä siksi, että useimmat nelipuun algoritmit perustuvat pisteen/tiedon etsiminen ja sen aikavaativuus on lineaarinen solmujen määrään verrattuna. (Samet ja Webber 1988, s. 55)

Kolmiulotteiselle datalle käytetään nelipuun sijasta kasipuuta, joka on nelipuun kolmiulotteinen yleistys. Samet (1984, s. 230) mainitsee, että ongelma useampiulotteisissa pistepohjaisissa tietorakenteissa (mm. kasipuu) on, että ne vievät niin paljon muistia.

3.3 Muut puumaiset laajennokset, joita käytetään spatiaalisen tiedon esittämisessä

On olemassa myös muita puumaisia tietorakenteita, joilla voidaan esittää spatiaalista tietoa. Esitellään niistä lyhyesti sekä k-d -puu, että R-puu.

3.3.1 K-d -puu

(Bentley 1975, s. 509) esittelee kehittämänsä k-d puun, joka on moniulotteinen binääripuu tietorakenne tiedon etsimiselle käyttäen assosiatiivistä hakua. (Bentley 1975, s. 516) mukaan k-d -puun sovelluskohteita ovat mm. karttasovellukset, jossa esimerkkinä voidaan etsiä mitkä kaupungit ovat tietyin koordinaattipisteen lähettyvillä ja puheentunnistuksen, jolloin lähimmän naapurin algoritmilla, voidaan päätellä mikä sana oli kyseessä (Bentley 1975, s. 516). Lähimmän naapurin algoritmin määritteleminen ei kuulu tähän tutkielmaan.

K-d -puun määritelmässä kirjain k tarkoittaa puun ulottovuuksien määrää. K-d -puun solmulla P on k -kappaletta avainarvoja $(K_0(P), ..., K_{k-1}(P))$ (Bentley 1975, s.510). Jokaisella solmulla on erottaja joka on kokonaisluku 0 ja $k-1$ välillä. Erottaja merkitään jokaiselle puun tasolle. Juurisolmun erottaja on aina 0 ja sen lasten 1. Tätä jatketaan kunnes ollaan lehtisolmujen tasolla. Kun saavutetaan erottajan arvo $k-1$, niin aloitetaan taas 0:sta. (Bentley 1975, s. 510).

Solmulla on myös kaksi osoitinta, jotka osoittavat solmun lapsisolmuihin. Bentley 1975, s. 510 on nimennyt osoittimet nimillä $HISON(P)$ ja $LOSON(P)$ sekä jakajan nimellä $DISC(P)$. Bentley 1975 määrittelee k-d -puun järjestyksen seuraavasti: "Jokaiselle solmulle P k-d -puussa, olkoon j on $DISC(P)$, tällöin kaikille solmuille $Q \in LOSON(P)$, on totta, että $K_j(Q) < K_j(P)$. Vastaavasti jokaiselle solmulle $R \in HISON(P)$, on totta, että $K_j(R) > K_j(P)$" (Bentley 1975, s. 510, suomennos minun).

Erottajan arvolla saadaan syvyydellä j tietää, että solmun P *HISON* puolella olevien solmujen ja niiden lapsisolmujen j avainarvo on suurempi kuin tarkasteltavan solmun P (Bentley 1975, s. 510).

(Samet 1984, s. 230-) pitää k-d puuta parannuksena pistepohjaiselle nelipuulle, koska sillä pystytään rajoittamaan puun oksien määrää ja siten puun leveyttä. K-d -puulla on mahdollista luoda vastine nelipuulle arvolla $k = 2$. Tällöin kyseessä olisi 2-ulotteinen binääripuun kaltainen puurakenne.

3.3.2 R puut

Guttmanin (1984, s. 47) mukaan alueellinen tieto on vaikeaa esittää pistetietona, tästä johtuen Guttman esittelee tekstissään ratkaisuksi moniulotteiselle datajoukolle (esimerkiksi maan alue kartalla) tietorakanteeksi R puuta. Nämä moniulotteiset alueet voivat olla esimerkiksi tallennettuna alueelliseen tietokantaan siten, että ne ovat tupleja, joista toinen arvo on indeksi ja toinen itse alue. (Guttman 1984, s. 48). R puu on tasapainoinen puu joka muistuttaa binääripuuta. R puun lehtisolmuissa on indeksit, jotka osoittavat alueellisen tietokannan dataobjektiin, sekä tieto N-ulotteisesta alueesta jonka sisälle kaikki lehtisolmun indeksin alueet mahtuu (Guttman 1984, s. 48). Solmuilla, jotka eivät ole lehtisolmuja on indeksit lap-

sisolmuihinsa ja vastaavasti tieto myös alueesta jonka se kattaa. Tämä alue sisältää kaikkien solmun lapsisolmujen alueet (Guttman 1984, s. 48).

(Guttman 1984, s. 48) määrittelee, että R puulla jonka solmulla on maksimissaan M kappaletta merkintöjä (lapsisolmuja tai indeksejä) ja olkoon $m \leq \frac{M}{2}$, jolla kuvataan merkintöjen minimimäärää. Tällöin R puulla on Guttmanin mukaan seuraavat ominaisuudet (Guttman 1984, s. 48, määritelmän ominaisuuksien suomennokset minun):

1. Jokaisen lehtisolmun sisältämä arvojen määrä on väliltä $m - M$, paitsi jos solmu on juurisolmu
2. Jokaisen lehtisolmun määrittämä alue on pienin mahdollinen joka sisältää indeksin määrittelemän datajoukon
3. Jokaisella ei-lehtisolmulla on minimissään m ja maksimissaan M lasta, ellei se ole juurisolmu
4. Jokasella ei-lehtisolmulla on määriteltynä pienin mahdollinen alue, joka sisältää kaikkien solmn lapsisolmujen määrittelemät alueet.
5. Juurisolmulla on vähintään kaksi lasta, ellei se ole lehtisolmu
6. Kaikki lehtisolmut ovat samalla tasolla

Avainluvuiksi Guttman kertoo R puulle, että R puun, jossa on N kappaletta eri datajoukkoja, korkeus on korkeimmillaan $log_m(N) - 1$. Solmujen maksimimäräksi Guttman kertoo, että tälläisessä puussa on $|\frac{N}{m}| + |\frac{N}{m^2}| + ... + 1$. Guttmanin mukaan pahimmassa tapauksessa yksittäisen solmun, joka ei ole juurisolmu, tilan hyödyntäminen on $\frac{m}{M}$ solmulle varatusta tilasta. Eli huonoimmassa tapauksessa varataan käyttämätöntä tallennustilaa solmua kohden jopa $1 - \frac{m}{M}$ verran.

3.4 Neli- ja kasipuun sovelluskohteita

Neli- ja kasipuita voidaan hyödyntää laajasti eri sovelluskohteissa. Niitä sovelletaan kuvien- ja videonpakkaamisen lisäksi mm. lääketieteellisissä kuvissa, astrofysiikassa, sosiaalisen median verkostoissa ja 3d-mallinnuksessa. Seuraavaksi esitellään esimerkit jokaiselle luetelluista käyttökohteista.

3.4.1 Lääketieteelliset kuvat

Lääketieteellisten kuvien pakkauksessa on erityisen tärkeää, ettei tietoa katoa. Pienikin tiedon puute saattaa vaikuttaa diagnoosin tekemiseen, joka voi aiheuttaa vakavia terveydellisiä tai laillisia ongelmia (Munteanu ym. 1999, s.176).

Kuvan, joka ei menetä yhtään pikseliä tallennuksessa, lähetyksessä tai latauksessa, siirto voi olla hyvinkin hidasta kuvassa esiintyvän tiedon määrän vuoksi. Erääksi vastaukseksi tähän ongelmaan on esitetty aallokemuunnos pohjaisia pakkausalgoritmeja. Tälläisissä algoritmeissä käytetään hyväksi nelipuun operaatioita. Perusidea näissä algoritmeissa on, että kuva siirretään vähän kerrallaan progressiivisesti jatkuvasti tarkemmaksi. Tarkoittaen, että ensin tulee huonolaatuinen kuva jota algoritmi parantaa ajan kanssa kokoajan tarkemmilla ja tarkemmilla yksityiskohdilla. Tällä tekniikalla kuvaa voidaan tarkentaa siihen pisteeseen, että kuvan paukkaussuhde on häviötön. Paras hyöty tästä saadaan kun tarkennetaan vain pientä ja kiinnostavaa osaa kuvasta, jolloin koko tiedonsiirtokaistaa voidaan käyttää tärkeään tietoon. (Schelkens ym. 2003, s. 441). (Munteanu ym. 1999, s. 176).

(Munteanu ym. 1999, s. 178) ehdottaa nelipuiden hyödyntämistä aallokemuunnos algoritmissä. Lyhyesti, nelipuu -sijoittelu jakaa kuvan kaksiulotteiseen homogeenisiin paloihin. Jako tehdään niin monta kertaa, että jaettava palanen on täysin homogeeninen tai sitten ollaan pienimmässä mahdollisessa palasessa (Munteanu ym. 1999, s. 178).

Dilmaghani ym. 2004 (s. 806) mukaan progressiivinen kuvanlähetys on tehokkaampaa interaktiivisissa kommunikaatiossa, jossa lähetetään kuvia ja jota käydään hitaassa verkossa. Lisäksi Dilmaghani ym. 2004 (s. 809) kertoo yhteenvedossa, että progressiivisten kuvan välityksellä on kaksi selvää hyötyä. Ensiksi jokaisella kierroksella vastaanottaja saa korkearesoluutioisen arvion kuvasta, joka koostuu nelipuun alipuun juurisolmuista, sen sijaan että saisi vain osan kuvasta. Edellä mainittua voisi havainnollistaa siten, että ensin lähetetään vastaava kuva, kuin kuvion 1(b) ja sen jälkeen seuraavassa aallossa tarkempi 1(a). Toiseksi, jos tulee tilanne että täyden resoluution kuva ei ole pakollista saada, niin tehokkuus kasvaa suuresti.

3.4.2 3D-tilan mallinnus

3D-tilan mallinnus on tärkeää mm. robotiikassa. Hornung ym. 2013 (s. 1) mukaan useimmat robotiikan applikaatiot tarvitsevat todennäköisyysesitystä, vapaan- tunnetun- ja tuntemattoman tilan mallinnusta, sekä sen lisäksi ajonaikaista tehokkuutta ja muistinkäytön tehokkuutta.

Avoimen lähdekoodin C++ kielen OctoMap kirjastoa voidaan käyttää 3d-tilan mallinnuksessa. Kyseisessä kirjastossa tehokkuuden saavuttamisen apuna on käytetty kasipuuta (Hornung ym. 2013, s. 1). Hornung ym. 2013, (s. 4) kirjoittaa, että koska kasipuut of hierarkkinen tietorakenne voidaan kasipuurakenne katkaista mistä kohdasta vain jolloin saavutetaan karkeampi jako kunhan sisemmistä solmuista huolehditaan. Tehokkuus tulee esiin siinä, että kyseisessä kirjastossa jokaisella solmulla voi olla vain kaksi eri arvoa. Ne ovat joko vapaita tai varattuja. (Hornung ym. 2013, s. 6).

3.4.3 Sosiaalisen median verkostot

Sosiaalisen median verkostoissa nelipuu-tietorakennetta soveltuu apuvälineeksi datan louhintaan. Nelipuihin voidaan esimerkiksi kategorisoida sosiaalisen median twiittejä niiden geo-merkinnän mukaan. Tämän tiedon avulla voidaan esimerkiksi etsiä kaikki twiitit tietystä aihealueesta, jotka on tehty tietyltä alueella (Jiang ym. 2015, s. 267).

Aluksi (Jiang ym. 2015, s. 271) määrittelee, että jokainen nelipuun lapsisolmu enkoodataan 2 bittisellä koodilla (00 ylävasen, 10 yläoikea, 11 alaoikea, 01 alavasen). Jonka jälkeen Jiang ym. 2015 luo alueelle oman geotiivisteen käyttämällä hyväksi nelipuita siten, että tiiviste luodaan nelipuun indekseistä menemällä haluttun tarkkuden syvyydelle nelipuuhun aina valitsemalla oikea lapsikoodi. Esimerkiksi kahden syvyisessä geotiiviste olisi 0001 jos haluttu alue on alueen ylävasen lohko (00) ja sitten alavasen (01). Seuraavaksi tämä saatu tiiviste muutettaisiin binääriluvusta 32-järjestelmän luvuksi (johon kuuluvat luvut 0-9 sekä kirjaimet a-z. Esimerkkinä (Jiang ym. 2015, s. 271) on käyttänyt koordinaatteja (-23.994140625, -46.23046875), josta on saatu nelipuun 20 bitin tarkkuudella 11001111111011010100, josta sitten yllä kerrotulla muunnoksella 32 järjestelmän luvuksi "6gxp".

3.4.4 Astrofysiikka

Astrofysiikan alalla neli- ja kasipuita hyödynnetään mm. eri avaruuden kappaleiden kartoittamisessa. Aversa ym. 2003, (s. 3) mukaan Particle-Particle ja Barnes-Hut algoritmit ovat melko yleisesti käytössä tähtien, galaksien ja tähtijoukkojen simuloinnissa. Näistä Barnes-Hut algoritmi käyttää hyväkseen neli- ja kasipuita. Kyseiset puurakenteet tulevat esiin siten, että toisistaan kaukana olevien tähtijoukkojen massakeskipisteet asetetaan neli- tai kasipuuhun ja sitten käydään läpi lehtisolmu lehtisolmulta ja lasketaan tähtien toisiinsa vaikuttavia voimia. Sitten lasketaan tähdille kiihtyvyydet ja paikanmuutokset ja ne asetetaan uudelleen neli- tai kasipuuhun ja toteutetaan laskut alusta. Nelipuusta saadaan hyöty siten, että ei välttämättä tarvitse laskea jokaista lehtisolmua vasten sitä kiihtyvyyttä, vaan jos on tiheä tähtijoukko, niin voidaan käyttää näiden yhdistävää solmua. Tämä auttaa suorituskyvyn puolesta huomattavasti (Aversa ym. 2003, s. 3-4).

Yksi kosmologian tutkimuskohteista on löytää kosmisia tyhjiöitä (cosmic void). Kosminen tyhjiö tarkoittaa aluetta, jonka alueella ei esiinny paljoa avaruuden materiaaleja (Gómez-Vargas, Medel-Esquivel ja García-Salcedo 2019). Gómez-Vargas, Medel-Esquivel ja García-Salcedo 2019 myös ehdottaa kasipuita yhdeksi vaihtoehdoksi kosmisten tyhjiöiden löytymiseksi. Kasipuita käytettäisiin jakamalla avaruuden kolmiulotteinen alue lehtisolmuihin ja tutkimalla syntyneitä lehtisolmuja. Jos solmu on tyhjä tai täynnä jätetään se siihen, muuten jaetaan alue uudestaan kahdeksaan lehtisolmuun (Gómez-Vargas, Medel-Esquivel ja García-Salcedo 2019, s. 4).

4 Videonpakkaus

Videonpakkausmenetelmiä on useita HEVC:n lisäksi. mm. HEVC:n edeltäjä AV1 ja seuraaja VVC. Lisäksi on myös kilpailevia menetelmiä kuten esimerkiksi Googlen tukema VP9. Valitsin tutkielmassa tarkemmin tutustuttavaksi menetelmäksi HEVC videonpakkausmenetelmän, koska se on moderni menetelmä, joka käyttää nelipuita hyödykseen pakkauksessa.

4.1 HEVC Videonpakkaus

HEVC (High Efficience Video Coding) standardi (H.265) on videonpakkaus standardi, jota on kehittäneet ryhmät ITU-T Video Coding Experts Group ja ISO/IEC Moving Picture Experts Group. HEVC kehitettiin korvaamaan silloiset videonpakkausmenetelmät, kuten H.264/MPEG-4 ja AVC (Sullivan ym. 2012, s. 1649). HEVC tavoitteena on 50% parannus bitinsiirtonopeuteen näihin verrattuna. (Sullivan ym. 2012, s. 1649). Ensimmäinen versio HEVC standardista otettiin käyttöön ITU-T puolesta 13.4.2013 ((ITU-T) 2020).

Videokuva koostuu sarjasta kuvia. Jokainen kuva jaetaan neliönmuotoisiin CTU (coding tree unit) palasiin, joiden koko vaihtelee 16x16, 32x32 ja 64x64 kuvapisteen välillä (Sullivan ym. 2012, s. 1650-1651). CTU koostuu samasta kuvasta luodusta kirkkauden CTB (coding tree block) ja kahdesta erillisistä väri CTB:ista. Jokainen CTU vastaa yksittäistä nelipuun juurta. Videon enkoodauksessa CTU jaetaan tarvittaessa rekursiivisesti neljään osaan CB:ksi (coding block), kunnes saavutetaan asetettu maksimisyvyys. CU (coding unit) koostuu yhdestä kirkkaus ja tavallisesti kahdesta väri CB:sta (Sullivan ym. 2012, s. 1651).

Pakkauksessa seuraavaksi CU:ille luodaan ennustus palat käyttämällä inter tai intra ennustusmalleja. Ennustusmallissa on tarkoitus luoda nimensä mukaisesti ennustus videon tarkasteltavasta kuvasta. Inter ennustuksessa verrataan nykyistä kuvaa sitä edeltävään ja mahdollisesti siitä seuraavaan kuvaan, kun taas intra ennustuksessa verrataan muuhun samassa kuvassa olevaan tietoon esimerkiksi viereisen CTU:n tai CU:n arvoihin (Sullivan ym. 2012, s. 1651). CTU:lle voidaan käyttää myös Skip ennustetta, mikä tarkoittaa, että kyseisen palasen ennustus ohitetaan (Sullivan ym. 2012, s. 1661). Videon ensimmäiselle kuvalle tehdään pelkkä intra ennustus (Sullivan ym. 2012, s. 1650). Inter tai intra ennustuksessa CB voidaan

edelleenjakaa jopa neljään osaan neljäksi PB:ksi (prediction block) (Sullivan ym. 2012, s. 1655). Vastaavasti kuin CU kohdalla, myös PU (prediction unit) koostuu kirkkauden ja kahden värin PB:ista (Sullivan ym. 2012, s. 1655).

Seuraavaksi otetaan jäännöskuva, joka saadaan alkuperäisen kuvan ja ennustetun kuvan erotuksesta. Jäännöskuvaa varten CB:t voidaan jakaa edelleen TB:iksi (transform block) (Sullivan ym. 2012, s. 1655-1656). Jäännöskuva otetaan, jotta voidaan lähettää/tallentaa vain tieto muutoksista, joka on vähemmän kuin alkuperäisen kuvan lähettäminen, verrattuna edelliseen kuvaan. Jäännöskuvasta muodosteteaan jäännösnelipuu. Jäännösnelipuu jaetaan vastaavasti kuin aikaisempi kuva TB:iksi (Sullivan ym. 2012, s. 1661).

Edellisestä operaatiosta saatu jännösnelipuu lähetetään koodekin muutos (transform) vaiheeseen, jolla pyritään saamaan esille lähetettävän tiedon toisteisuutta. Lopuksi data siirretään itse datan enkoodaukseen, jossa suoritetaan videonpakkaus (Sullivan ym. 2012, s. 1652).

5 Yhteenveto

Tämä kandidaatin tutkielma on kirjallisuuskatsaus, jossa tarkoituksena oli tutkia miten nelipuuta käytetään videonpakkauksessa. Tutkielmassa tutustuttiin häviölliseen ja häviöttömään tiedonpakkausmenetelmään ja niiden eroihin, kuten että nimensä mukaisesti häviöllisessä pakkauksessa on mahdollista, ettei alkuperäistä dataa saada enään pakkauksen jälkeen palautettua ja häviöttömän pakkauksen tehottomuuteen verrattuna häviölliseen pakkaukseen.

Lisäksi tutkielmassa tutustuttiin puurakenteisiin ja niistä eritoten nelipuuhun ja sen variantteihin, kuten R- ja k-d -puu, sekä laajennoksiin kuten kasipuu. Esitellessä nelipuun (ja kasipuun) sovelluskohteita kävi selväksi, että puurakenteita käytetään laajasti eri sovellusaloilla ja niillä on paikkansa nykypäivän sovelluskehityksessä. Tiedonpakkaus on isossa osassa nykypäivän tiedonsiirtoa ja nelipuista saadaan jo itsessään hyötyä tiedon pakkaamisessa. Nelipuita käyttäviä uusia algoritmeja ja tutkimuksia tehdään laajasti ja tästä aihealueesta löytyykin lähteitä ja tutkimustietoa monelta eri vuosikymmeneltä.

HEVC videonpakkauksessa nelipuut ovat isossa roolissa ja nelipuita käytettiinkin apuna kun luotiin ennustus paloja videon kuville. Videonpakkaus onkin yksi yleisimmistä nelipuun käyttökohteista ja Zhang ja Mao 2019 (s. 19-20) pitääkin todennäköisinä tulevaisuuden trendeinä videonpakkaamisessa myös koneoppimista, fyysisen laitteen ohjelmiston kiihdytystä (Hardware acceleration), uusia videonpakkausta vahvasti käyttäviä sovelluksia sekä energiatietoisia videonpakkaus standardeja. Zhang ja Mao 2019 (s. 18) mukaan, VVC:tä pidetään yhtenä seuraavista videonpakkaus standardeista alalla.

Lähteet

(ITU-T), Telecommunication Standardization Sector. 2020. “ITU-T Recommendations”. Viitattu 9. marraskuuta. https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=11885&lang=en.

Aversa, Alan G, ym. 2003. “Astrophysics and Cosmology Special Interest Group “Galaxy Simulations””. http://www.u.arizona.edu/~aversa/galaxysims.pdf.

Bentley, Jon Louis. 1975. “Multidimensional Binary Search Trees Used for Associative Searching”. *Commun. ACM* (New York, NY, USA) 18, numero 9 (syyskuu): 509–517. ISSN: 0001-0782. doi:10.1145/361002.361007. https://doi.org/10.1145/361002.361007.

Bing, Benny. 2015. “Next-Generation Video Coding and Streaming”. Luku Video coding fundamentals, toimittanut Benny Bing. John Wiley & Sons, Incorporated, 2015. ProQuest Ebook Central. https://ebookcentral.proquest.com/lib/jyvaskyla-ebooks/detail.action?docID=4406551.

Dilmaghani, R.S., A. Ahmadian, M. Ghavami ja A.H. Aghvami. 2004. “Progressive medical image transmission and compression”. *IEEE Signal Processing Letters* 11 (10): 806–809. doi:10.1109/LSP.2004.835563.

Gómez-Vargas, Isidro, Ricardo Medel-Esquivel ja Ricardo García-Salcedo. 2019. “Cosmic voids, spatial algorithms and data structures”. *Journal of Physics: Conference Series* 1221 (kesäkuu): 012031. doi:10.1088/1742-6596/1221/1/012031. https://doi.org/10.1088%5C%2F1742-6596%5C%2F1221%5C%2F1%5C%2F012031.

Guttman, Antonin. 1984. “R-Trees: A Dynamic Index Structure for Spatial Searching”. Teoksessa *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data,* 47–57. SIGMOD ’84. Boston, Massachusetts: Association for Computing Machinery. ISBN: 0897911288. doi:10.1145/602259.602266. https://doi.org/10.1145/602259.602266.

Hornung, A., K. M. Wurm, M. Bennewitz, C. Stachniss ja W. Burgard. 2013. “OctoMap: an efficient probabilistic 3D mapping framework based on octrees”. *Autonomous Robots* 34 (3): 189–206. https://doi.org/10.1007/s10514-012-9321-0.

Huffman, David A. 1952. “A Method for the Construction of Minimum-Redundancy Codes”. *Proceedings of the IRE* 40 (9): 1098–1101. doi:10.1109/JRPROC.1952.273898.

Hunter, G. M., ja K. Steiglitz. 1979. “Operations on Images Using Quad Trees”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1, numero 2 (huhtikuu): 145–153. ISSN: 1939-3539. doi:10.1109/TPAMI.1979.4766900. https://ieeexplore.ieee.org/document/4766900.

Jiang, Jinling, Hua Lu, Bin Yang ja Bin Cui. 2015. “Finding Top-k Local Users in Geo-Tagged Social Media Data”. *2015 IEEE 31st International Conference on Data Engineering:* 267–278. https://ieeexplore.ieee.org/abstract/document/7113290.

Munteanu, A., Cornelis J., Van Der Auwera G. ja P. Cristea. 1999. “Wavelet Image Compression — The Quadtree Coding Approach”. *IEEE Transactions on Information Technology in Biomedicine* 3 (3): 176–185. https://ieeexplore.ieee.org/abstract/document/788579.

Samet, H., ja R. E. Webber. 1988. “Hierarchical data structures and algorithms for computer graphics. I. Fundamentals”. *IEEE Computer Graphics and Applications* 8, numero 3 (toukokuu): 48–68. ISSN: 1558-1756. doi:10.1109/38.513.

Samet, Hanan. 1980. “Deletion in Two-Dimensional Quad Trees”. *Commun. ACM* (New York, NY, USA) 23, numero 12 (joulukuu): 703–710. ISSN: 0001-0782. doi:10.1145/359038.359043. https://doi.org/10.1145/359038.359043.

———. 1984. “The Quadtree and Related Hierarchical Data Structures”. *ACM Comput. Surv.* (New York, NY, USA) 16, numero 2 (kesäkuu): 187–260. ISSN: 0360-0300. doi:10.1145/356924.356930. https://doi.org/10.1145/356924.356930.

———. 1985. “Data Structures for Quadtree Approximation and Compression”. *Commun. ACM* (New York, NY, USA) 28, numero 9 (syyskuu): 973–993. ISSN: 0001-0782. doi:10.1145/4284.4290. https://doi.org/10.1145/4284.4290.

Sayood, K. 2006. *Introduction to Data Compression.* EBSCO ebook academic collection. Elsevier Science. ISBN: 9780126208627. https://books.google.fi/books?id=044wLaqZ8twC.

Sayood, Khalid. 2017. *Introduction to data compression.* s.2. Morgan Kaufmann.

Schelkens, P., A. Munteanu, J. Barbarien, M. Galca, X. Giro-Nieto ja J. Cornelis. 2003. "Wavelet coding of volumetric medical datasets". *IEEE Transactions on Medical Imaging* 22, numero 3 (maaliskuu): 441–458. ISSN: 1558-254X. doi:10.1109/TMI.2003.809582.

Shih, Frank Y. 2010. "Image Processing and Pattern Recognition: Fundamentals and Techniques". Luku Image representation and description, toimittanut Frank Y. Shih. Piscataway, New Jersey : IEEE Press c2010. https://ieeexplore.ieee.org/book/5559001.

Sonka, Milan, Vaclav Hlavacc ja Roger Boyle. 1999. *Image Processing, Analysis, and Machine Vision.* 511 Forest Lodge Road, Pacific Grove, CA 93950, USA: Brooks/Cole Publishing Company.

Sullivan, G. J., J. Ohm, W. Han ja T. Wiegand. 2012. "Overview of the High Efficiency Video Coding (HEVC) Standard". *IEEE Transactions on Circuits and Systems for Video Technology* 22, numero 12 (joulukuu): 1649–1668. ISSN: 1558-2205. doi:10.1109/TCSVT.2012.2221191.

Zhang, Ticao, ja Shiwen Mao. 2019. "An Overview of Emerging Video Coding Standards". *GetMobile: Mobile Comp. and Comm.* (New York, NY, USA) 22, numero 4 (toukokuu): 13–20. ISSN: 2375-0529. doi:10.1145/3325867.3325873. https://doi.org/10.1145/3325867.3325873.