Pintoja kuvaavien verkkojen muodostaminen ja optimointi

Antti Seppälä

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Pro gradu -tutkielma
Joulukuu 2001

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Antti Seppälä: Pintoja kuvaavien verkkojen muodostaminen ja optimointi.
Pro gradu -tutkielma, 63 sivua, 3 liitesivua

Joulukuu 2001

Tässä työssä tutkitaan pintojen kuvaamista kolmiulotteisten verkkojen avulla, verkkojen muodostamista automaattisesti ja niiden optimointia. Verkonmuodostusmenetelmissä painopiste on sellaisissa algoritmeissa, jotka eivät aseta vaatimuksia syötteen muodolle tai järjestykselle. Verkonmuodostusalgoritmeja tutkittiin käytännössä power crust -algoritmilla ja Hoppen algoritmilla. Näitä testattiin geneerisillä ja lääketieteellisillä syötteillä. Käytetty lääketieteellinen syöte muodostetiin magneettiresonanssiaivokuvien pohjalta. Power crust -algoritmi tuotti huomattavasti monimutkaisempia verkkoja kuin Hoppen algoritmi. Magneettiresonanssidatan kanssa päästiin parhaaseen lopputulokseen power crust -algoritmin avulla. Hoppen algoritmi tuottaa geometrisesti tasapainoisempia verkkoja, mutta ei kykene toistamaan yhtä yksityiskohtaisia pintoja kuin power crust -algoritmi.

Avainsanat: Pinnan kuvaaminen verkolla, kolmiulotteinen verkko, verkon generointi, verkon optimointi

Kiitokset

Tämä työ on rahoitettu Tekesin Lääke2000 -teknologiaohjelmaan kuuluvan projektin *Functional brain maps – Enchancement of early clinical drug development* saamasta rahoituksesta ja Tampereen yliopiston Tietojenkäsittelytieteiden laitoksen stipendinä. Lisäksi haluan kiittää ohjaajiani Jouni Mykkästä ja Jussi Tohkaa asiantuntevasta ja kannustavasta ohjauksesta.

Sisällys

1. Johdanto

Tässä työssä tutkin pinnan kuvaamista kolmiulotteisten verkkojen avulla, verkkojen tuottamista automaattisesti sekä niiden optimointia. Kolmiulotteista kuvadataa tuotetaan hyvin erilaisilla menetelmillä. Esimerkiksi useat lääketieteelliset instrumentit, tekniset mittauslaitteet sekä tietokonesimulaatiot tuottavat lopputuloksenaan kolmiulotteista kuvadataa jossakin muodossa. Kolmiulotteisen mittaustuloksen tulkitseminen vaatii usein sen muokkaamista ja muuttamista ihmisen helpommin ymmärtämään muotoon. Erityisesti silloin, kun tuloksen tulkitseminen perustuu visuaaliseen havainnointiin. Joukko näytearvoja ei sellaisenaan anna katsojalle käsitystä tutkittavasta kappaleesta, vaan näytteet olisi kyettävä visualisoimaan ihmisen ymmärtämässä muodossa. Tähän tarvitaan menetelmiä, joilla voidaan muodostaa kolmiulotteisia malleja annetun näytejoukon perusteella.

Kolmiulotteisten pintojen esittämistä ja käsittelyä on tutkittu tietokonegrafiikan ja laskennallisen geometrian alalla jo pitkään. Samoin erilaisia menetelmiä mallien muodostamiseen näytejoukon perusteella on tunnettu jo 1980-luvulta lähtien. Varhaisimmat menetelmät olivat kuitenkin tarkoitettu jossakin tietyssä muodossa annetun syötteen käsittelyyn. Nämä algoritmit käyttivät hyväkseen syötealkioiden järjestystä tai jotakin muuta syötteen sisältämää tietoa tutkittavasta pinnasta. Kolmiulotteisen kuvadatan yleistyessä syntyi tarve kehittää ratkaisuja, jotka eivät perustu tiettyyn datan muotoon tai tuotantotapaan. Vasta viime vuosina tässä on onnistuttu tyydyttävästi. Tällaiset yleiset verkonmuodostusmenetelmät kykenevät tuottamaan mallin täysin mielivaltaisesta näytteestä, pelkästään käyttämällä apunaan näytepisteiden koordinaatteja.

Pinnan esitystavaksi on tässä työssä valittu monikulmioverkko ja erityisesti kolmioista rakentuva verkko. Kaikki tutkitut algoritmit ovat sellaisia, että ne tuottavat lopputuloksenaan monikulmioverkon. Tietokonegrafiikassa monikulmioverkko on klassinen pinnan esitystapa ja sen esittäminen ja käsittely on teknisesti helppoa. Kolmiulotteista esittämistä tukevat laitteistot on usein optimoitu juuri kolmioverkkojen piirtämiseen.

Verkkojen tuottamisen lisäksi ongelmaksi on muodostunut kuvauksen koon ja laadun optimointi. Mittauslaitteet tuottavat usein niin tarkkaa tietoa kohteesta, että sen perusteella rakennetuista malleista tulee hyvin monimutkaisia, usein liian monimutkaisia niiden käyttötarkoitukseen nähden. Automaattisesti muodostettujen mallien geometria ei useinkaan ole optimaalinen. Ne saattavat esimerkiksi sisältää alueita, jotka voitaisiin

esittää täysin vastaavasti huomattavasti pienemmällä määrällä monikulmioita. Tätä varten tarvitaan menetelmiä, joilla verkkoa voidaan optimoida. Tavoite on muodostaa verkko, joka on mahdollisimman yksinkertainen, mutta kuvaa silti mahdollisimman hyvin haluttua pintaa.

Työn tavoitteena on tutkia yleispäteviä verkonmuodostusmenetelmiä ja verkon optimointimenetelmiä. Yleispätevällä tarkoitetaan algoritmia, joka ei vaadi syötteeltään tiettyä muotoa tai järjestystä. Tarkoitus on selvittää millaisia lähestymistapoja ja menetelmiä tällaisissa algoritmeissa käytetään ja kokeilla niiden toimivuutta käytännössä sekä testidatalla, että todellisilla näytteillä. Samalla vertaillaan eri lähestymistapojen tuottamien verkkojen laatua.

Tutkielma on jaoteltu seuraavasti. Luvussa 2 esitellään muutamia peruskäsitteitä, joita tarvitaan myöhemmissä luvuissa. Määrittelen ensinnäkin mikä on kolmiulotteinen verkko ja miten sillä kuvataan pintoja. Lisäksi esittelen muutamia hyödyllisiä laskennallisen geometrian käsitteitä, kuten keskiakselimuunnos, Voronoi-diagrammi ja Delaunay-kolmiointi. Erityisesti kahdella viimeksi mainitulla on keskeinen rooli useissa verkonmuodostusmenetelmissä. Luvussa 3 käsittelen kolmiulotteisten verkkojen muodostamista automaattisesti. Luvussa 4 tutkin verkon optimointia. Käsittelen verkon optimoinnin periaatteita ja tavoitteita yleisesti sekä otan lähempään tarkasteluun kaksi erilaista lähestymistapaa. Luku 5 on tutkielman käytännön osuus. Luvuissa 3 ja 4 esiteltyjä menetelmiä kokeiltiin käytännössä ja saatuja verkkoja vertailtiin keskenään. Tutkimusaineistona käytettiin sekä magneettiresonanssikuvien perusteella tuotettuja syötteitä, että geneerisesti tuotettua kolmiulotteista dataa.

2. Kolmiulotteiset verkot ja niihin liittyviä käsitteitä

Käytän tutkielmassa monikulmiosta myös termiä *polygoni* ja monikulmioverkosta termiä *polygoniverkko*. Monikulmion sanotaan olevan *konveksi*, jos minkään sen särmän kautta piirretty suora ei leikkaa muita sen särmiä. Tätä voi havainnollistaa seuraavalla esimerkillä: leikataan monikulmion malli pahvista ja venytetään sen ympärille kuminauha. Monikulmio on konveksi mikäli kuminauhan ja pahvimallin reunan väliin ei missään kohtaa jää tyhjää tilaa. Konveksisuuden käsite toimii vastaavasti myös kolmannessa ulottuvuudessa monitahokkaiden suhteen. Kolmiulotteista pintaa kuvaavaa verkkoa kutsun yksinkertaisesti *kolmiulotteiseksi verkoksi*.

2.1 Pintojen kuvaaminen kolmiulotteisilla verkoilla

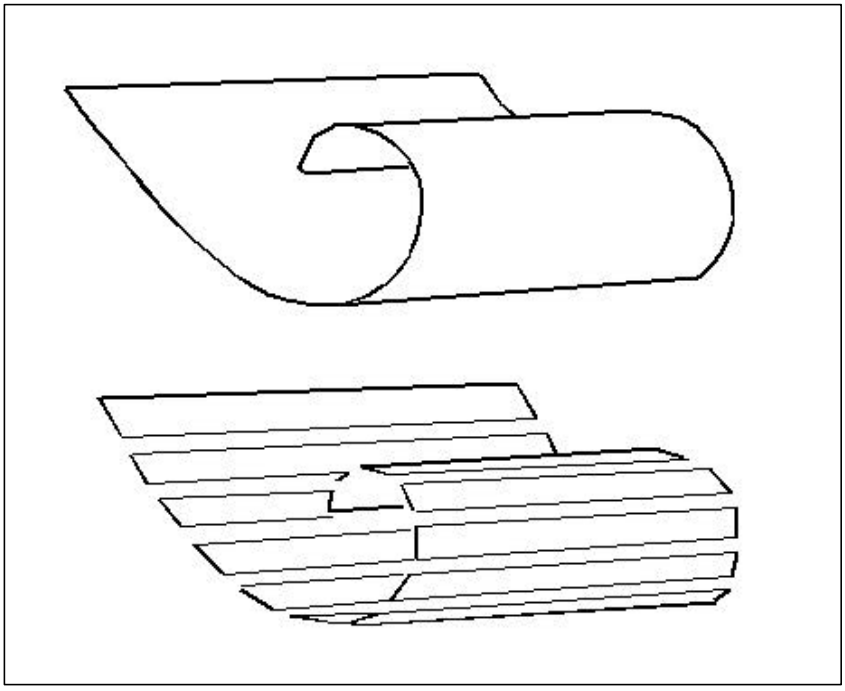
Polygoniverkko (polygon mesh) on kokoelma kärkiä, särmiä ja niistä rakentuvia monikulmioita. Intuitiivisesti ajatellen se on joukko reunoistaan toisiinsa liitettyjä monikulmioita, jotka yhdessä kuvaavat pintaa. Tällaisesta pinnan esityksestä käytetään termiä *paloittain lineaarinen pinta* (piecewise-linear surface). Sana "rautalankamalli" antaa jonkinlaisen käsityksen, mistä on kysymys. Verkon osista käytetään myös termiä *simpleksi* (simplex). Yksinkertaisin simpleksi on kärki. Särmä rakentuu kahdesta kärjestä ja monikulmio joukosta särmiä. Kaikki muut simpleksit paitsi kärki rakentuvat siis joukosta hierarkiassa alempana olevia simpleksejä.

Polygoniverkko sisältää tietoa verkon itsensä topologiasta ja verkon kuvaaman pinnan geometriasta. Topologia kertoo miten monikulmiot rakentuvat kärkien ja särmien avulla ja miten ne liittyvät toisiinsa. Geometria taas kertoo millainen on se kappale tai pinta jota verkon avulla kuvataan. Muodollisesti polygoniverkko koostuu joukosta avaruuden pisteitä ja relaatiosta näiden pisteiden välillä. Se voidaan määritellä kaavalla $M = (V, K)$ [Hop94], missä V on pistejoukko ja K simplekseistä rakentuva systeemi, joka määrittelee miten kärjet, särmät ja monikulmiot liittyvät toisiinsa. Systeemi K rakentuu joukosta simpleksejä, joita kutsutaan kärjiksi. Lisäksi siihen kuuluu joukko näiden kärkien yhdistelmänä muodostettuja simpleksejä. Jokainen kärkeä monimutkaisempi simpleksi voidaan jakaa osiin siten, että jokainen ei-tyhjä, systeemiin K kuuluvan simpleksin osajoukko kuuluu myös systeemiin K. Muotoa $\{i, j\} \in K$ olevia simpleksejä kutsutaan särmiksi ja muotoa $\{i, j, k, ...\} \in K$ olevia simpleksejä tahoiksi tai monikulmioiksi. Systeemi K ilmaisee, että on olemassa tietynlainen

verkko, jossa tietyt kärjet muodostavat särmiä ja nämä särmät tietyllä tavalla monikulmioita. Se ei kuitenkaan kerro mitään siitä millaista kappaletta tai pintaa verkko mahdollisesti kuvaa, eikä edes sitä, missä avaruudessa se sijaitsee. Vasta kun otetaan mukaan pistejoukko V ja liitetään se systeemiin K, niin tiedetään millaista pintaa verkko kuvaa. Joukon V sanotaan *realisoivan* verkon johonkin avaruuteen. Samaan systeemiin K voidaan liittää useita pistejoukkoja jotka kaikki realisoivat erilaisen verkon, mahdollisesti vielä eri ulotteisiin avaruuksiin.

Polygoniverkot ovat klassinen, muttei suinkaan ainoa tapa kuvata kolmiulotteista pintaa. Pinta voidaan kuvata esimerkiksi käyttämällä matemaattisia funktioita, kuten beziér- tai NURBS -pintoja [Wat93]. Niiden hyvä puoli on, että kappaleen kuvaus saadaan usein mahtumaan pieneen tilaan ja kuvaus vastaa tarkasti kuvattavaa pintaa. Tällainen esitystapa myös säilyttää tarkkuutensa riippumatta siitä miltä etäisyydeltä pintaa tarkastellaan. Matemaattisesti täsmällisen esitysmuodon käyttäminen pinnan kuvaamiseen on kuitenkin usein varsin epäkäytännöllistä ja kaikissa tapauksissa sillä ei saavuteta etua polygoniverkkoihin nähden. Useimmat visualisointiin tarkoitetut laitteistot eivät kykene esittämään suoraan funktioilla kuvattua pintaa, vaan kuvaus on ennen piirtämistä muutettava jonkilaiseksi approksimaatioksi, useimmiten polygoniverkoksi. Myöskään täsmällisen esitystavan tarjoamasta tarkkuudesta ei aina ole hyötyä. Automaattisesti tuotetun verkon lähteenä käytetty pistejoukko on yleensä saatu ottamalla kohdekappaleesta joukko näytteitä. Koska näytteitä on rajallinen määrä, niin osa kohdekappaleen informaatiosta väistämättä menetetään. Vaikka näytteenä saadun pistejoukon perusteella muodostettaisiinkin matemaattisesti täsmällinen kuvaus niin se ei useinkaan kuvaisi alkuperäistä kappaletta yhtään sen paremmin kuin jokin vähemmän tarkka esitysmuoto. Pistejoukon perusteella voidaan vain arvioida miltä alkuperäinen kappale näyttää. Tällaisen arvion esittämiseen polygoniverkko on riittävä väline.

Polygoniverkko on vain harvoin pinnan tarkka kuvaus. Tämä johtuu siitä, että verkon rakennuspalikat, eli monikulmiot, ovat tasojen osia. Esimerkiksi aidosti kaarevaa pintaa on mahdoton kuvata tarkasti tasojen osista koostuvalla esityksellä, sillä tämä vaatisi, että monikulmioita on äärettömän suuri määrä ja että ne ovat äärettömän pieniä. Kuvauksen tarkkuus riippuu paitsi kuvaavasta verkosta, myös siitä millainen kuvattava pinta on. Jos pinta on valmiiksi planaarinen, eli koostuu vain tasojen osista, se voidaan esittää täsmällisesti polygoniverkolla. Useissa tapauksissa polygoniverkko on kuitenkin vain pinnan aproksimaatio, jonka tarkkuus riippuu kuvattavasta pinnasta ja polygonien koosta. Kuvassa 2.1 on esitetty kaarevan pinnan approksimointi tasojen avulla.



Kuva 2.1 Kaarevan pinnan approksimointi tasojen avulla

2.1.2 Kolmioista rakentuvat verkot

Eräs polygoniverkkojen erikoistapaus ovat *kolmioverkot* (triangular meshes). Nimensä mukaisesti tällaisen verkon kaikki monikulmiot ovat kolmioita. Kolmio on yksinkertaisin geometrinen primitiivi, jolla on pinta-ala, joten se sopii luonnostaan perusyksiköksi pintaa kuvaavaan esitykseen. Mikä tahansa monikulmio on mahdollista kuvata kolmioista rakentuvalla esityksellä, joten polygoniverkolla kuvattu muoto voidaan esittää aina myös kolmioverkolla. Jos verkossa käytetään vain kolmioita, ei verkon ilmaisuvoimaa rajoiteta, mutta tarvittavat tietorakenteet yksinkertaistuvat.

Kolmioverkoilla saavutetaan myös muita etuja. Esimerkiksi kaikki kolmion kärjet sijaitsevat automaattisesti samalla tasolla, sillä kolmion kärkipisteet määrittelevät tason. Jos monikulmioon lisätään neljäs kärki, ei voida tutkimatta olla varmoja sijaitseeko neljäs kärki samalla tasolla kuin kolme aiempaa. Jos se ei sijaitse, niin monikulmio joudutaan jakamaan kahteen kolmioon, sillä muuten neljän kärjen monikulmio ei enää muodosta tasoa. Tämä ei ole ongelma, jos varkossa käytetään pelkästään kolmioita. Lisäksi kolmiulotteiseen esittämiseen tarkoitetut laitteistot on yleensä optimoitu piirtämään nimenomaan kolmioita. Kolmioverkon huono puoli on, että se usein sisältää enemmän särmiä kuin vastaava rajoittamattomista monikulmioista koostuva verkko.

Kolmioverkon tilantarve ei välttämättä ole optimaalinen. Tässä työssä oletetaan, että verkko on kolmioverkko ellei toisin mainita.

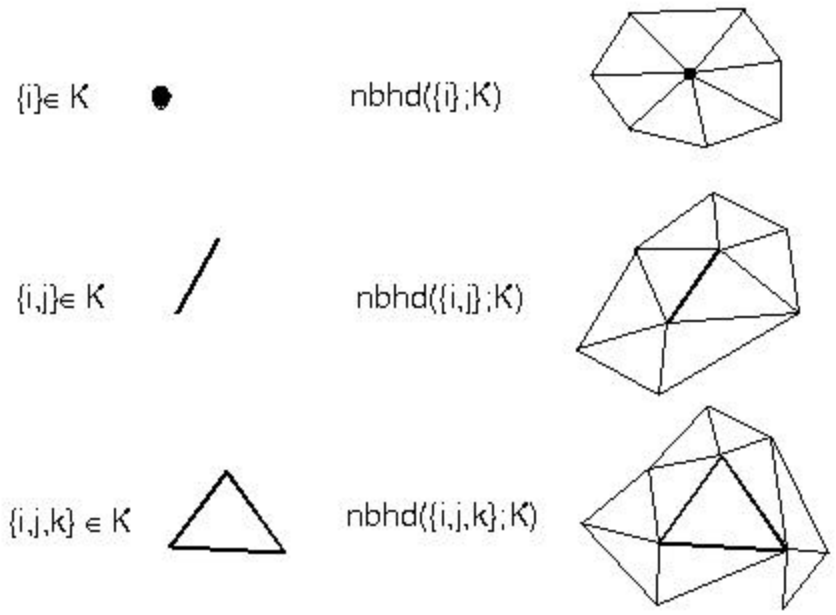
Eräs laskennallisen geometrian ongelma-alue on jonkin geometrisen muodon jakaminen pienempiin osiin. Yksinkertaisin yksikkö, johon tasojen osista rakentuva muoto voidaan jakaa on kolmio. *Kolmiointi* (triangulation) on kolmioista rakennettu esitys, joka vastaa muodoltaan alkuperäisen kappaleen tai pinnan muotoa. Termiä käytetään myös silloin, kun pistejoukosta rakennetaan kolmioverkko, niin että joukon pisteet ovat kolmioiden kärkinä.

2.1.3 Simpleksien ympäristöt

Hyödyllinen verkkoihin liittyvä käsite on simpleksien ympäristöt. Tässä työssä käytetty määrittely on peräisin Hoppelta [Hop94]. Simpleksin s ympäristöön kuuluvat kaikki ne verkon simpleksit, joihin s jotenkin liittyy. Esimerkiksi kärjen ympäristöön kuuluvat kaikki ne särmät ja kolmiot, joita muodostamassa kärki on mukana. Muodollisesti simpleksin ympäristö määritellään seuraavasti: Olkoon $J \subset K$ joukko verkon simpleksejä. Nyt joukon J ympäristö $nbhd(J;K)$ määritellään seuraavasti

$$nbhd(J;K) = \{s \in K : \exists s'' \in J, s' \in K \rightarrow s'' \cup s \subset s'\}. \qquad (1.1)$$

Kärjen, särmän ja kolmion ympäristöt on esitetty kuvassa 2.2.



Kuva 2.2 Simpleksien ympäristöt

2.1.4 Verkon laadun mittaaminen

Verkon laadun määrittäminen on vaikea tehtävä ja riippuu siitä, mihin tarkoitukseen verkkoa tullaan käyttämään ja mitkä ovat ensisijaiset verkolle asetettavat vaatimukset. Jokaista kappaletta kohti on olemassa ääretön määrä verkkoja, joita voidaan pitää sen paloittain lineaarisena vastineena. Miten näistä verkoista valitaan paras tai miten yleensäkään määritellään mikä niistä on hyvä? Aikaisemmin pintaa kuvaavan verkon laatua on luonnehdittu hyvin subjektiivisilla tavoilla, esimerkiksi arvioimalla silmämääräisesti miten hyvin verkko kuvaa alkuperäistä pintaa. Vasta viime vuosina [Alb95, Dom98, Knu01] on alettu tutkia ja kehittää täsmällisesti määriteltyjä metriikoita verkon laadun kuvaamiseen. Nämä antavat kuitenkin vain kyvyn mitata tiettyjä verkon ominaisuuksia, eivätkä anna yleispätevää hyvän verkon määritelmää. Tapauskohtaisesti täytyy edelleen ratkaista mitä verkolta halutaan.

Verkon käsittelyn kannalta on hyvä, jos se olisi mahdollisimman yksinkertainen. Geometriselta kannalta äärimmilleen yksinkertaistettu verkko ei enää välttämättä kuvaa kovin hyvin alkuperäistä pintaa. Yksi laadun kriteeri on monikulmioiden muoto. Tavoite saattaa esimerkiksi olla, että verkon monikulmiot ovat muodoltaan mahdollisimman lähellä tasasivuista kolmiota. Toisissa tilanteissa, esimerkiksi nesteanalyysissä, saatetaan haluta, että monikulmiot ovat pitkiä ja kapeita. Yksinkertaisin ja tärkein metriikka on verkon monimutkaisuus, eli kärkien, särmien ja kolmioiden lukumäärä. Jos kaksi mallia näyttävät silmämääräisesti samoilta, mutta toisessa on huomattavasti enemmän kolmioita kuin toisessa, niin monimutkaisempi malli saattaa olla kyseiseen tilanteeseen liian monimutkainen. Tilanne muuttuu, jos katseluetäisyyttä muutetaan lähemmäksi, jolloin monimutkaisemmassa mallissa saattaa erottua yksityiskohtia, joita yksinkertaisemmassa mallissa ei ole. Tällöin yksinkertaisempaa mallia voidaan pitää tilanteeseen nähden huonompana, paitsi jos tärkeimmäksi kriteeriksi on asetettu esimerkiksi kuvaruudun päivitysnopeus. Hyvän verkon määritelmä riippuu siis täysin tilanteesta ja vaadituista kriteereistä. Yleisesti ottaen tavoitteena on kuitenkin saada aikaan sopiva tasapaino yksinkertaisuuden ja tarkkuuden välillä. Verkon laatua kuvaavia metriikoita tarvitaan apuna myös verkkoa optimoitaessa, sillä muuten olisi mahdotonta päätellä onko optimointiprosessi menossa oikeaan suuntaan.

2.2 Verkkojen muodostamiseen ja käsittelyyn liittyviä menetelmiä

Seuraavaksi esittelen muutamia verkkojen muodostamisen yhteydessä käytettyjä matemaattisia ja geometrisia menetelmiä.

2.2.1 Eulerin karakteristika

Hyödyllinen polygoniverkkoihin liittyvä väline on Eulerin karakteristika. Se lasketaan seuraavasti

$$E_k = V - E + F \quad . \tag{1.2}$$

Kaavassa V on verkon kärkien lukumäärä, E verkon särmien lukumäärä ja F verkon monikulmioiden lukumäärä. Eulerin karakteristikalla saadaan tietoa verkon kuvaaman kappaleen topologiasta. Se antaa arvon 2 jos verkko on topologialtaan pallomainen. Pallomaisella topologialla tarkoitetaan pintaa, joka voidaan muuttaa palloksi pelkästään venyttämällä sen osia. Esimerkiksi kuutio on topologialtaan pallomainen, mutta kahvikuppi ei ole, koska siinä on kahva.

2.2.2 Pinnan suunnan määrittäminen

Verkkoja ja pintoja käsiteltäessä tarvitaan usein tietoa pinnan suunnasta jossakin tietyssä kohdassa tai tietyllä alueella. Esimerkiksi verkkoa visualisoitaessa halutaan usein jättää piirtämättä ne pinnan osat, jotka ovat katsojasta poispäin. Samoin valaistuksen ja varjostusten laskemisessa tarvitaan tietoa pinnan suunnasta. Tähän tarkoitukseen käytetään normaalivektoreita. Pinnan normaalivektori on vektori, joka on kohtisuorassa pintaa vasten.

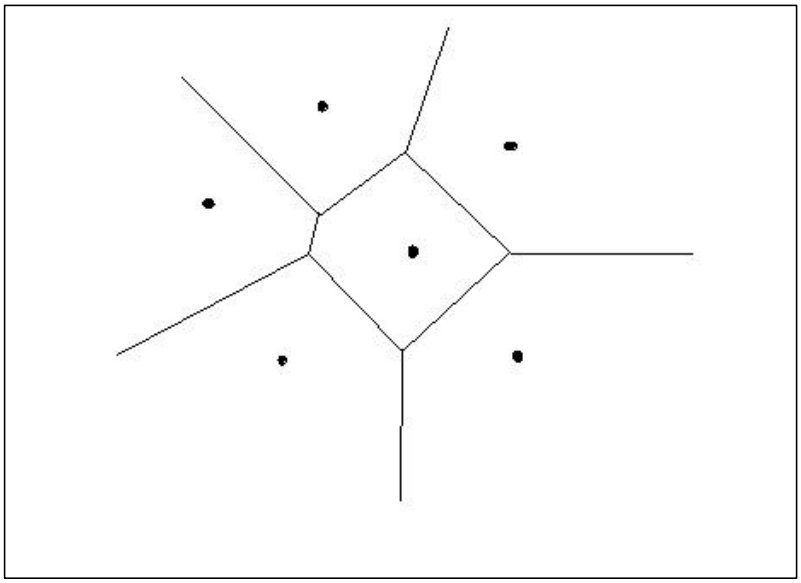
Normaalivektorien laskeminen liittyy usein yhtenä vaiheena kolmiulotteisten verkkojen muodostamiseen. Verkkojen yhteydessä normaalivektorit muodostetaan yleensä kolmioille ja kärjille. Koska kolmio määrittelee tason, on kolmion normaalivektori samalla myös sen tason normaalivektori jolla kolmio lepää. Tässä työssä normallivektoreita tarvitaan erityisesti verkon optimointiin tarkoitettujen menetelmien yhteydessä.

2.2.3 Voronoi-diagrammi

Oletetaan että on olemassa joukko pisteitä ja valitaan tästä joukosta satunnaisesti jokin piste. Rajataan sen jälkeen valitun pisteen ympäriltä tietty alue siten, että rajatun alueen jokainen kohta on lähempänä valittua pistettä kuin mitään toista joukon pistettä. Jos näin rajattu alue on suurin mahdollinen tällainen alue sitä kutsutaan *Voronoi-vyöhykkeeksi* (Voronoi region) tai *Voronoi-soluksi* (Voronoi cell). Esitystä, joka kuvaa pistejoukon kaikkien pisteiden Voronoi-solut kutsutaan *Voronoi-diagrammiksi*.

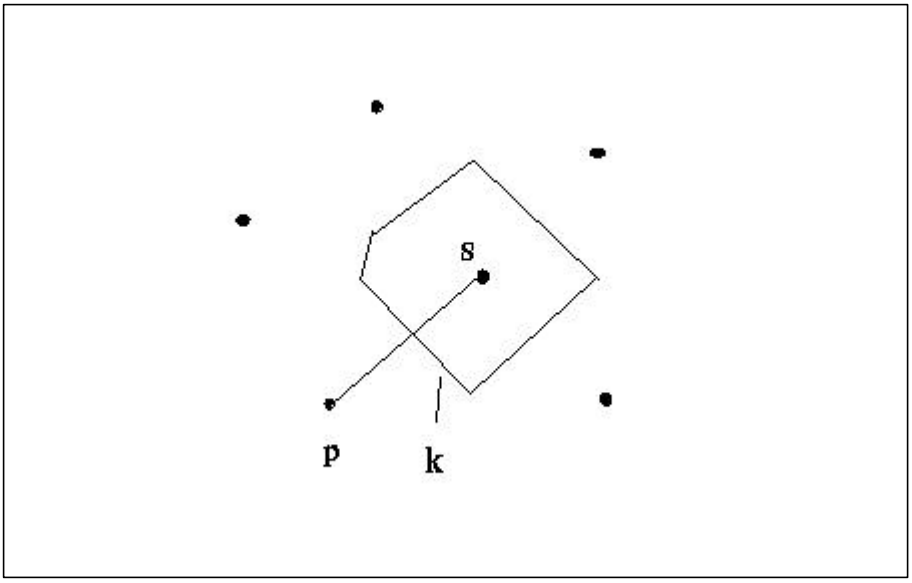
Muodollisesti sama voidaan esittää seuraavasti. Olkoon $S \subseteq R^n$ (joukon S alkioista käytetään usein kirjallisuudessa termiä "site"). Valitaan piste $s \in S$ ja muodostetaan joukko V_s siten, että $V_s = \{v \in R^n \mid p_i \in S, dist(s,v) < dist(p_i,v), p_i \neq s\}$ missä $dist(X,Y)$ kertoo pisteiden X ja Y välisen etäisyyden. Nyt joukko V_s on pisteen s Voronoi-solu.

Pistettä, jossa Voronoi-solujen särmät kohtaavat kutsutaan *Voronoi-kärjeksi* (Voronoi vertex). Kolmiulotteisessa Voronoi-diagrammissa jokainen Voronoi-kärki kuuluu vähintään neljään Voronoi-soluun. Näihin soluihin liittyvät pistejoukon pisteet ovat kyseisen kärjen lähimmät naapurit. Voronoi-palloksi sanotaan palloa, jonka keskipiste on jokin Voronoi-kärki ja joka kulkee kärjen lähimpien naapureiden kautta. Kuvassa 2.3 on esitettu kaksiulotteinen pistejoukko ja sen Voronoi-diagrammi. Sama voidaan laajentaa myös kolmanteen ulottuvuuteen. Tällöin Voronoi-solu ei ole kaksiulotteinen monikulmio vaan kolmiulotteinen monitahokas. Voronoi-solu voi olla muodoltaan joko avoin tai suljettu. Suljetut Voronoi-solut ovat aina muodoltaan konvekseja.



Kuva 2.3 Pistejoukko ja sen Voronoi-diagrammi

Laskennallisessa geometriassa Voronoi-diagrammi on hyödyllinen apuväline ja sitä käytetään myös muutamassa tässä työssä esitellyssä menetelmässä. Sen avulla voidaan esimerkiksi nopeuttaa tilanteita, jossa halutaan etsiä jokin pisteen lähimmät naapurit tai yleisesti tilanteita, joissa tarvitaan tietoa siitä, millainen jonkin pisteen lähiympäristö on.



Kuva 2.4 Voronoi-solun muodostaminen.

Kuvassa 2.4 on esitetty osa kaksiulotteista Voronoi-diagrammia. Kuvaan on merkitty piste *s* ja sen Voronoi-solu, sekä pistettä *s* lähinnä sijaitsevat joukon pisteet. Yhtä niistä merkitään tunnuksella *p*. Tarkastellaan Voronoi-solun särmää *k*. Jos vedetään jana pisteiden *s* ja *p* väliin niin havaitaan, että särmä *k* leikkaa tämän janan täsmälleen sen puolivälissä ja kulkee kohtisuorassa janaa vasten. Voidaan valita minkä tahansa Voronoi-solun särmä ja todeta, että pistejoukosta löytyy aina sellainen piste, johon piirretyn janan valittu särmä leikkaa sen puolesta välistä. Tämän huomion avulla Voronoi-solun rakentaminen on suoraviivaista. Etsitään ensin pistettä s lähinnä oleva piste ja muodostetaan suora joka kulkee kohtisuoraan näiden kahden pisteen muodostaman janan puolessa välissä. Näin saatu suora jakaa avaruuden kahteen osaan, jolloin piste *s* jää toiseen osaan ja lähinnä sijaitseva piste toiseen osaan. Sitten etsitään seuraavaksi lähin piste ja muodostetaan sille vastaavanlainen suora. Näin jatketaan, kunnes saadut suorat rajaavat suljetun konveksin alueen tai suorien rajaamalla alueella ei ole enempää joukon pisteitä.

Edellä kuvattu menetelmä on suoraviivainen, mutta ei erityisen tehokas. Sen avulla Voronoi-diagrammi voidaan muodostaa ajassa $O(n^2 \log n)$. Paras tällä hetkellä tunnettu algoritmi muodostaa Voronoi-diagrammin ajassa $O(n \log n)$ ja se tunnetaan nimellä Fortunen algoritmi [For95]. Voronoi-diagrammin laskeminen voidaan palauttaa *n*-alkioisen numerosarjan järjestämiseen, jonka aikavaatimus on pahimmassa tapauksessa juuri $O(n \log n)$. Fortunen algoritmi on siis algoritmisessa mielessä optimaalinen.

2.2.4 Delaunay-kolmiointi

Jos tarkastellaan pistejoukon Voronoi-diagrammia ja piirretään viiva niiden pisteiden välille, joiden Voronoi-solut koskettavat toisiaan, niin saadaan lopputuloksena kolmiointi. Tällainen kolmiointi tunnetaan nimellä Delaunay-kolmiointi.

Delaunay-kolmioinnin keksi alunperin venäläinen geometrikko Boris Delaunay. Hän havaitsi, että tietynlaisella kolmioinnilla on erityisiä geometrisia ominaisuuksia. Voronoi-diagrammin ja Delaunay-kolmioinnin välillä vallitsee läheinen suhde, sillä kuten edellä esitettiin, Voronoi-diagrammi sisältää implisiittisesti Delaunay-kolmioinnin. Delaunay-kolmiointi on Voronoi-diagrammin geometrinen duaali.

Kaksiulotteisen Delaunay-kolmioinnin erottaa muista joukon mahdollisista kolmioinneista seuraava ominaisuus: jos otetaan mikä tahansa kolmioinnin kolmio ja piirretään ympyrä, joka kulkee sen kärkien kautta, niin voidaan olla varmoja siitä että ympyrän alueella ei ole muita joukon pisteitä kuin valitun kolmion kärjet. Delaunay-kolmiointi on aina olemassa äärelliselle pistejoukolle ja se on yksikäsitteinen. Sama voidaan laajentaa kolmanteen ulottuvuuteen. Tällöin kolmioinnin perusyksikkö ei kuitenkaan ole kaksiulotteinen kolmio, vaan kolmiulotteinen tetraedri. Kolmiulotteinen Delaunay-kolmiointi on joukko tetraedrejä, joiden kärkinä on pistejoukon pisteitä. Sama tyhjän ympyrän -sääntö on kuitenkin edelleen voimassa. Tetraedrin kärkien kautta kulkevan pallon sisälle ei jää muita joukon pisteitä, kuin tetraedrin kärjet.

2.2.5 Kolmiulotteinen kuvadata

Digitaalisen kaksiulotteisen kuvan pienintä rekenneyksikköä kutsutaan tunnetusti *pikseliksi* (pixel, picture cell). Vastaavasti kolmiulotteisen kuvadatan pienintä rakenneyksikköä kutsutaan *vokseliksi* (voxel, volume cell). Se voidaan mieltää pieneksi kuutioksi tai suorakulmaiseksi särmiöksi. Vokseli on kolmiulotteisen kuvadatan pienin jakamaton elementti.

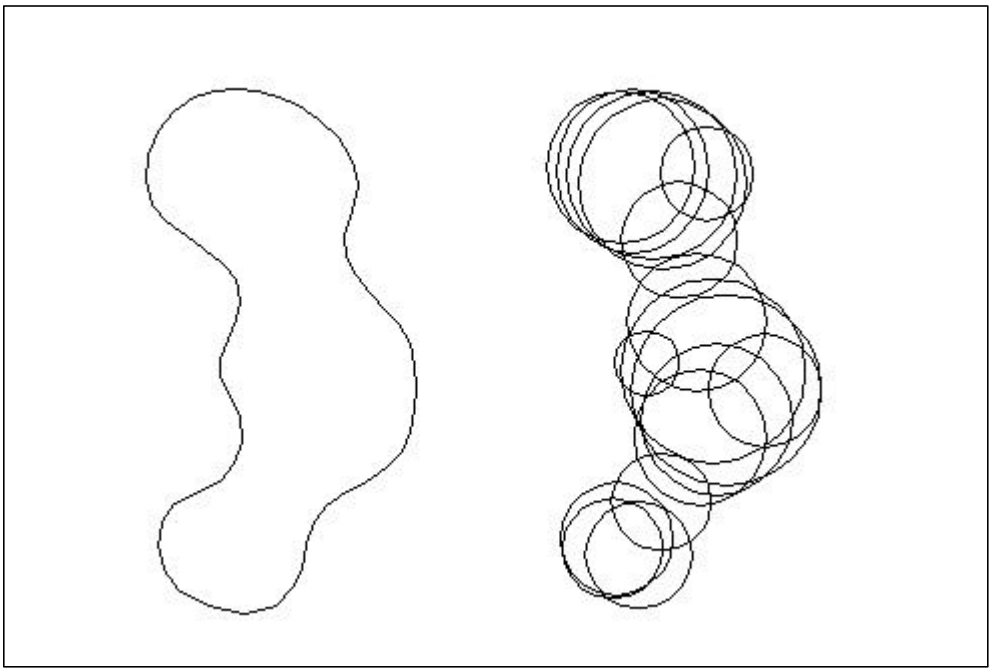
Vokseli sisältää intensiteettiarvon, joka kuvaa sen kirkkautta. Magneettiresonanssikuvissa intensiteettiarvo kertoo mihin kudostyyppiin vokseli kuuluu. Käsittelemätön vokselidata sisältää useita kudostyyppejä ja pintoja. Tällaisen datan segmentointi ja tiettyyn pintaan kuuluvien vokselien etsiminen siitä on kokonaan oma tutkimusalansa, eikä kuulu tämän työn piiriin. Tässä työssä esitellyissä menetelmissä oletetaan, että syötedata sisältää vain yhden pinnan

2.2.6 Keskiakselimuunnos

Keskiakselimuunnos (medial axis transform) on eräs tapa kuvata alueen tai tilan muoto. Se kehitettiin alunperin biologian ja lääketieteen tarpeisiin ja se solveltuukin hyvin epäsäännöllisten ja orgaanisten muotojen kuvaamiseen. Myöhemmin sitä on hyödynnetty myös laskennallisen geometrian ja tietokonegrafiikan alueella.

Määritellään käsite *tyhjä pallo* tarkoittamaan sellaista palloa, joka on kokonaan kappaleen sisä- tai ulkopuolella, eli kappaleen pinta ei leikkaa palloa. *Maksimaalinen tyhjä pallo* on sellainen tyhjä pallo, joka koskettaa kappaleen pintaa vähintään kahdesta kohdasta. Kappaleen keskiakselimuunnos on maksimaalisten tyhjien pallojen joukko. Näiden pallojen keskipisteet muodostavat kappaleen keskiakselin, jota kutsutaan myös *luurangoksi* (skeleton). Muunnoksen tarkkuus riippuu siitä, kuinka monta maksimaalista tyhjää palloa joukkoon otetaan, täydellisessä muunnoksessa palloja on ääretön määrä. Eräs kaksiulotteinen keskiakselimuunnos on esitetty kuvassa 2.5.

Keskikselimuunnos on hyvin kompakti tapa kuvata kappaleen muoto, sillä se koostuu vain joukosta keskipisteitä ja säteitä. Sen muodostaminen on kuitenkin hankala ja laskennallisesti vaativa operaatio. Voronoi-diagrammin avulla voi muodostaa keskiakselimuunnoksen arvion.



Kuva 2.5 Kaksiulotteinen muoto ja sen keskiakselimuunnos

3. Verkkojen muodostaminen automaattisesti

Tässä luvussa tutkitaan, miten pintaa kuvaavia verkkoja voidaan muodostaa automaattisesti kolmiulotteisten näytteiden perusteella. Ongelma on osa laajempaa tutkimusaluetta, jota kutsutaan *pinnan uudelleenmuodostukseksi* (surface reconstruction). Sen piiriin kuuluvat kaikki ne tilanteet, joissa pinnasta saatujen näytteiden perusteella halutaan muodostaa matemaattinen kuvaus alkuperäisestä pinnasta. Hoppe [Hop92] luokittelee pinnan uudelleenmuodostukseen kehitetyt menetelmät kahteen ryhmään sen perusteella millaisen kuvauksen ne muodostavat pinnasta uudelleenmuodostuksen aikana. Nämä luokat ovat *implisiitiset* ja *parametriset uudelleenmuodostusmenetelmät.* Implisiittistä pinnan esitystapaa käsitellään tarkemmin myöhemmin tässä työssä. Tämä jaottelu ei kerro missä muodossa algoritmin antama lopputulos esitetään, vaan ainoastaan sen, minkä tyyppisellä lähestymistavalla menetelmä yrittää ratkaista pinnan uudelleenmuodostuksen ongelman. Sekä implisiittisistä että parametrisistä pinnan kuvauksista voidaan lopulta muodostaa monenlaisia pinnan esitystapoja. Tässä työssä rajoitutaan niihin menetelmiin, joissa lopullinen pinnan esitystapa on polygoniverkko.

Kolmiulotteisia näytteitä syntyy hyvin monenlaisten prosessien seurauksena, esimerkiksi lääketieteellisen magneettiresonanssi- ja positroniemissiotomografiakuvauksen sekä erilaisten etäisyyden mittaamiseen perustuvien skannausmenetelmien (laserskannerien) avulla. Ihmisen on hyvin vaikea hahmottaa kappaletta pelkän pistemäisen tiedon perusteella. Näytteen analysointia helpottaa, jos sen perusteella voidaan muodostaa kolmiulotteinen malli. Koska näytteitä tuotetaan hyvin vaihtelevilla tavoilla, myös niiden sisältämä tietomäärä alkuperäisen kappaleen pinnasta vaihtelee huomattavasti. Samoin vaihtelee näytteiden muoto ja esitystapa. Mallin tuottamiseen tietyn muodon ja esitystavan omaavista näytteistä on kehitetty lukuisia ratkaisuja. Paras ratkaisu olisi kuitenkin menetelmä, joka tuottaa mallin mahdollisimman monenlaisista syötteistä. Hoppe muotoili ensimmäisenä ongelman abstraktilla tasolla ja lähti etsimään yleistä ratkaisua, joka ei vaadi syötteeltä tiettyä järjestystä tai esitystapaa [Hop92]. Tällaiselle yleiselle verkonmuodostusalgoritmille syötteeksi riittää pelkkä järjestämätön joukko kolmiulotteisia pisteitä. Amenta, Bern ja Kamvysselis täydensivät myöhemmin ongelman määrittelyä [Ame98].

Täsmällisesti ilmaistuna ongelma on seuraava: haluamme joukosta kolmiulotteisia näytepisteitä muodostaa paloittain lineaarisen pinnan, joka riittävällä tarkkuudella vastaa geometrialtaan ja topologialtaan alkuperäistä pintaa. Tavoite on, että ratkaisu on mahdollisimman yleispätevä ja helppokäyttöinen. Ihannetapauksessa lopputuloksen laatuun vaikuttaa vain syötejoukon näytetiheys, eikä esimerkiksi käyttäjän tapauskoh-

taisesti määrittelemät parametrit. Yleispätevä verkonmuodostusalgoritmi ei voi olettaa näytteiden sisältävän minkäänlaista tietoa alkuperäisen pinnan muodosta, vaan pinnan topologia ja verkon geometria on kyettävä päättelemään pelkästään näytepisteiden koordinaattien perusteella. Tässä työssä alkuperäisen pinnan muodosta ei tehdä muita oletuksia, kuin että se ei saa leikata itseään.

Koska alkuperäisen pinnan muodolle ei saa asettaa muita kuin edellä kuvattuja vaatimuksia, niin algoritmien olisi aina kyettävä tuottamaan jonkinlainen järkevä lopputulos mikäli syötejoukko on otettu itseään leikaamattomasta pinnasta ja se on riittävän tiheä. Käytännössä algoritmit tekevät kuitenkin alkuperäisen pinnan topologiasta erilaisia oletuksia. Voidaan esimerkiksi olettaa, että pinta ei sisällä reikiä. Jotkut algoritmit saattavat tulkita reiät vain alueiksi, joissa näytetiheys on matalampi ja sulkea ne, kun taas toiset algoritmit pitävät riittävät suuria tyhjiä alueita reikinä ja jättävät ne tyhjäksi. Lopputulos ei aina vastaa kovin hyvin alkuperäistä pintaa, vaikka se olisikin "järkevä". Käytännössä tämä tarkoittaa sitä, että täysin yleispätevää verkonmuodostusalgoritmia ei voida kehittää vaan käyttäjän on aina tiedettävä jotain alkuperäisestä pinnasta ja käytetyn algoritmin toiminnasta.

Verkonmuodostukseen tarkoitetut algoritmit voidaan luokitella kahteen ryhmään sen mukaan, mitä vaatimuksia ne asettavat syötteenään saamalleen datalle.

1. *Yleiset verkonmuodostusalgoritmit* muodostavat verkon järjestämättömästä pistejoukosta, eivätkä aseta syötteelle mitään muita vaatimuksia. Pisteet saavat sijaita täysin satunnaisessa järjestyksessä, eikä syöte sisällä mitään muuta tietoa, kuin pisteiden koordinaatit.

2. *Järjestettyä- tai osittain järjestettyä syötettä käsittelevät algoritmit* olettavat, että syötejoukon pisteet on ennalta järjestetty jokin algoritmin toimintaa helpottavan kriteerin mukaan. Tätä järjestämistä ei välttämättä ole tehty tietoisesti, vaan data on sitä tuotettaessa saattanut luonnostaan saada jonkin järjestetyn muodon ja algoritmi on myöhemmin suunniteltu tätä erikoistapausta varten. Lisäksi syöte saattaa sisältää pisteiden koordinaattien lisäksi myös jotain muuta algoritmin toimintaa helpottavaa tietoa. Esimerkiksi laserskanneri voi tuottaa tietoa pinnan normaalivektorista näytepisteen kohdalla, koska tiedetään mistä suunnasta näyttepisteen havainnointi tapahtui.

Verkonmuodostusalgoritmeja voidaan vertailla niiden tehokkuuden perusteella tai vertailemalla niiden tuottaman lopputuloksen laatua. Kuten luvussa 2 esittettiin, lopputuloksen laadun arviointi ei aina ole yksikäsitteinen tehtävä, vaan riippuu siitä mitä vaatimuksia lopputulokselle asetetaan. Verkonmuodostusalgoritmia valittaessa käyttäjän onkin ensin selvitettävä sopivat valintakriteerit. Päävaatimus voi olla esimerkiksi algoritmin tehokkuus, lopputuloksen laatu tai syötteen formaatti. Yksi algoritmien vertailuperuste on myös se, miten hyvin ne selviytyvät tietynlaisista hankalista syötteistä. Tällaisia hankalia syötteitä ovat esimerkiksi erityisen vaikeista ja monimutkaisista muodoista otetut näytteet tai sellaiset syötteet joissa on kohinaa.

Tässä työssä painopiste on järjestämätöntä pistejoukkoa käsittelevissä yleisissä verkonmuodostusalgoritmeissa. Tavoite on löytää mahdollisimman yleispätevä ja helppokäyttöinen ratkaisu edellä kuvatut rajoitukset huomioon ottaen. Tutkitut algoritmit on valittu näistä lähtökohdista. Aluksi käsittelen verkon muodostamiseen implisiittisistä pinnoista. Osa tutkituista yleisistä verkonmuodostusalgoritmeista käyttää näitä menetelmiä apunaan. Käsittelen lyhyesti myös muutamia yleisten verkonmuodostusalgoritmien kehitykseen vaikkuttaneita menetelmiä. Luvun loppuosassa käsitellään yleisiä verkonmuodostusalgoritmeja.

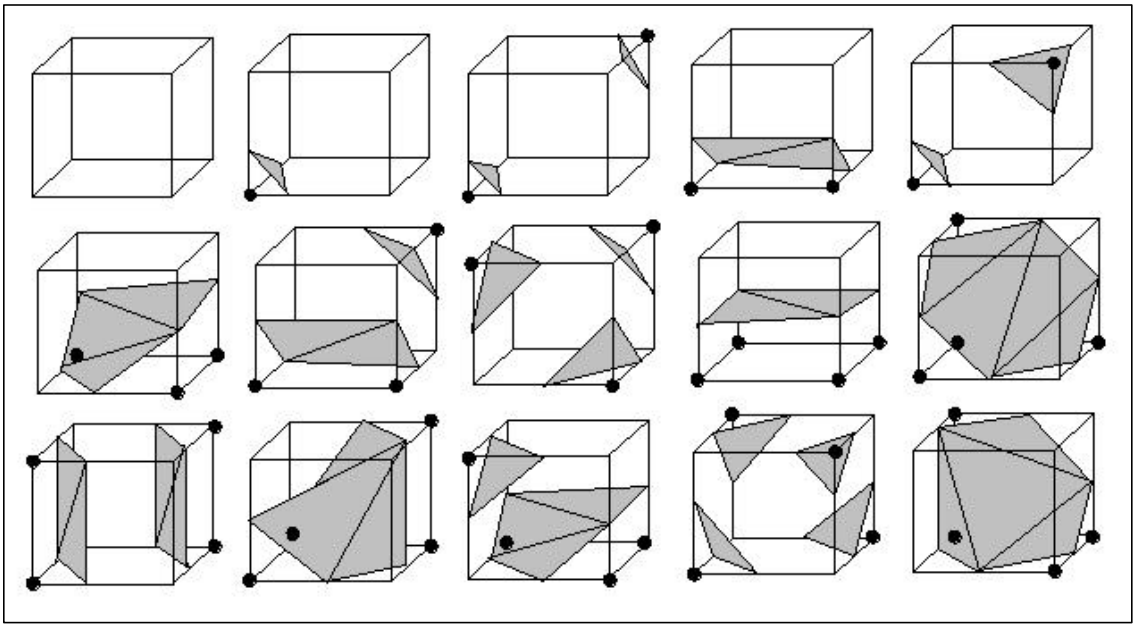
3.1 Verkon muodostaminen implisiittisistä pinnoista

Kaikkein eniten tutkittu verkonmuodostuksen alue on verkon muodostaminen vokselimuotoisesta datasta. Vokselidatan katsotaan kuuluvan *implisiittisiin pintoihin* (implicit surface). Implisiittisillä pinnoilla tarkoitetaan pintoja, jotka määritellän muotoa $f(x, y, z) = 0$ olevilla funktioilla. Yksinkertainen esimerkki on funktio $x^2 + y^2 + z^2 - r^2 = 0$, joka määrittelee pallopinnan säteellä r. Vokselidatan sisältämän pinnan voidaan ajatella olevan jonkin tällaisen funktion ratkaisujoukko, joka on pilkottu kuutiomaisiin paloihin. Menetelmistä, joilla muodostetaan verkkoja implisiittisistä pinnoista jakamalla pinnan sisältämä tila ensin kuutioihin, käytetään englanninkielisessä kirjallisuudessa termiä "*implicit surface tilers*" [Nin93].

Yleinen lähestymistapa näissä menetelmissä on jakaa tila jollakin tarkkuudella kuutioihin (ellei se ole jo valmiiksi jaettu, kuten vokselidatassa), tutkia mitkä kuutiot pinta leikkaa ja muodostaa sitten leikkautuvista kuutioista joukko monikulmioita. Tällaisista menetelmistä tunnetuin ja hyvin laajasti käytetty on Lorensenin ja Clinen [Lor87] *marching cubes* -algoritmi. Se on saavuttanut eräänlaisen referenssimenetelmän aseman ja muita implisiittisiä pintoja käsitteleviä algoritmeja verrataan monesti juuri siihen.

Marching cubes -algoritmi käy vuorotellen läpi kuvadatan vokselit ja selvittää mikä tarkasteltavan vokselin suhde on kuvattavaan pintaan. Vokseli mielletään kuutioksi, jolla on tietty tilavuus. Algoritmi tutkii onko tarkasteltava vokseli etsityn pinnan kohdalla ja jos on, niin mitkä kuution särmistä pinta leikkaa. Mikäli pinta kulkee vokselin läpi niin vokselista muodostetaan joukko kolmioita sen perusteella mitkä särmät leikkautuivat ja saadut kolmiot lisätään lopputuloksena muodostettavaan verkkoon. Sitten siirrytään seuraavaan vokseliin ja suoritetaan samat toimenpiteet sille. Lorensen ja Cline kutsuivat tätä siirtymistä "marssimiseksi", mistä algoritmin nimi on peräisin.

Kolmioiden muodostaminen vokselista tapahtuu sen perusteella, mitkä vokselin kulmapisteistä ovat pinnan sisäpuolella ja mitkä ulkopuolella. Kuvattava pinta leikkaa aina ne vokselin särmät, joiden päätepisteistä toinen on pinnan ulkopuolella ja toinen sisäpuolella. Koska vokselin kulmapisteellä voi olla kaksi tilaa (sisällä tai ulkona) ja koska vokselissa on 8 kulmapistettä on olemassa vain $2^8 = 256$ erilaista mahdollisuutta sille, mitkä vokselin särmät pinta voi leikata. Nämä mahdollisuudet voidaan numeroida ja tallettaa taulukkoon, jolloin algoritmin toteutuksessa on helppo selvittää leikkautuvat särmät muodostamalla binääriluku kulmapisteiden tilasta ja käyttämällä sitä indeksinä taulukkoon.



Kuva 3.1 Mahdolliset tavat, joilla pinta voi leikata kuution ja niiden tuottamat kolmioinnit.

Kun tiedetään mitkä kuution särmistä leikkautuvat, voidaan kuutiosta muodostaa kolmiointi. Jokainen mahdollinen tapa, jolla vokselin särmät voivat leikkautua tuottaa eri tyyppisen kolmioinnin. Jos mahdollisten kolmiointien joukosta poistetaan ne tapaukset, jotka ovat toistensa peilikuvia, niin erilaisia kolmiointityyppejä jää jäljelle vain 15. Ne on esitetty kuvassa 3.1. Ainoa laskettavaksi jäävä asia on se mistä kohtaa särmä leikkautuu.

Algoritmia on tutkittu runsaasti ja siitä on löydetty useita puutteita. Ning ja Bloomenthal [Nin93] osoittavat tutkimuksessaan erään ongelman, joka koskee muitakin kuution särmän leikkautumiseen perustuvia menetelmiä. On mahdollista, että kuution särmät leikkautuvat siten, että kuvan 6 esittämien vaihtoehtojen perusteella voidaan tuottaa useampi kolmiointi, jotka kaikki ovat periaatteessa oikein. Jos tällaisessa epävarmassa tilanteessa kahden vierekkäisen kuution kohdalla valitaan eri vaihtoehto, niin verkkoon syntyy reikiä ja pinnan topologia muuttuu. Algoritmi 1 kuvaa marching cubes -algoritmin perusmuodossaan.

Marching cubes -algoritmista on kehitetty muunnelmia, joilla yritetään korjata alkuperäisen menetelmän tunnettuja heikkouksia. Yksi esimerkki on Montanin, Scatenin ja Scopignon *Discretized Marching Cubes, DiscMC* [Mon94]. Se eroaa alkuperäisestä algoritmista lähinnä siinä miten kuution särmän ja halutun pinnan leikkauspiste määritellään. Alkuperäisessä versiossa leikkauspiste laskettiin mahdollisimman tarkasti lineaarisesti interpoloimalla, mutta DiscMC

Algoritmi 1

Marching cubes –algoritmi

```
Syöte: joukko vokseleita
Tulos: kolmioverkko
    for jokaiselle vokselille do
        Aseta indeksi arvoon 0.
        for jokaiselle vokselin kärjelle do
            Tutki onko kärki pinnan sisä- vai ulkopuolella.
            if (kärki on pinnan sisäpuolella) do
                Sytytä indeksiin kärjen järjestysnumeroa vastaava bitti.
        if (indeksi ? 0 and indeksi ? 255) do    // Tutkitaan oliko vokseli kokonaan sisällä tai ulkona.
            Etsi indeksin avulla taulukosta sopiva särmien leikkautumismalli.
            Muodosta mallin perusteella kolmiointi ja lisää kolmiot verkkoon.
```

algoritmi olettaa aina, että leikkauspiste on leikkautuvan särmä keskipiste. Tällöin leikkauspisteen laskeminen nopeutuu ja mahdollisten leikkauspisteiden määrä saadaan rajattua kiinteäksi. Myös niiden tasojen määrä, joihin syntyneet kolmiot kuuluvat saadaan kiinteäksi. Huono puoli on, että tällöin algoritmin tarkkuus kärsii ja suurin virhe jonka menetelmä tuottaa on puolet kuution särmän pituudesta.

Toinen esimerkki implisiittisiä pintoja käsitteleviä menetelmistä on Hiltonin ja Illingworthin [Hil97] *marching triangles* –algoritmi. Ning ja Bloomenthal [Nin93] ovat julkaisseet tutkimuksen, jossa luokitellaan ja vertaillaan eri menetelmiä verkon muodostamiseen implisiittisistä pinnoista.

3.2 Yleisiä verkonmuodostusalgoritmeja edeltäneitä lähestymistapoja

Tässä kohtaa esittelen lyhyesti muutamia historiallisesti tärkeitä lähestymistapoja ja algoritmeja, joiden voidaan katsoa vaikuttaneen yleisten verkonmuodostusalgoritmien kehitykseen.

Usein viitattu lähestymistapa esiteltiin Edelsbrunnerin ja Mücken [Ede94] *alpha – hahmoja* (alpha shapes) käsitelleessä tutkimuksessa. Tutkimuksen yhteydessä luonnosteltiin menetelmä kolmiulotteisen Delaunay-kolmioinnin muodostamiseen. Siinä kolmiointi muodostetaan lisäämällä syötejoukon pisteet kolmiointiin yksi kerrallaan ja varmistamalla lisätyn pisteen lähiympäristön perusteella, että kolmionti säilyy haluttuna. Tätä varten pistejoukon pisteet täytyy ensin järjestää jonkin koordinaatin mukaan. Menetelmä on ensimmäisiä lähestymistapoja, joilla päästiin lähelle yleistä ratkaisua verkonmuodostukseen organisoimattomasta pistejoukosta [Hop94]. Ongelmaksi muodostuu kuitenkin menetelmän syötteelle asettamat rajoitukset. Jos syötejoukossa on kohinaa tai kuvattava pinta ei täytä tiettyjä vaatimuksia, niin lopputuloksena ei saada aikaan pintaa, vaan eräänlainen rajakerros, jolla on jokin paksuus.

Toinen Delaunay-kolmiointiin perustuva menetelmä on Boissonatin [Boi84] julkaisema algoritmi. Siinä kolmiulotteisen Delaunay-kolmioinnin sisältämistä tetraedreistä pyritään eliminoimaan yksi kerrallaan ne, jotka jotka eivät kuulu kappaleen pintaan. Tästä käytetään termiä *sculpting*. Samassa yhteydessä Boissonat esitteli myös toisen menetelmän pinnan etsintään. Se soveltuu tiettyihin erikoistapauksiin. Tässä menetelmässä ongelma pyrittiin palauttamaan kaksiulottoiseen tilanteeseen valitsemalla pistejoukosta

jonkin pisteen lähiympäristöä kuvaava osajoukko ja projisoimalla tämä joukko siitä muodostetulle tangenttitasolle. Tämä projektio voidaan sitten kolmioida jollakin sopivalla menetelmällä, esimerkiksi kaksiulotteisella Delaunay-kolmioinnilla. Osa menetelmän yhteydessä esitellyistä ideoista on hyvin samankaltaisia, kuin seuraavaksi käsiteltävässä Hoppen algoritmissa.

3.3 Hoppen algoritmi

Tälle algoritmille ei varsinaisesti ole annettu omaa nimeä, joten kutsun sitä tässä työssä kehittäjänsä mukaan. Hoppen algoritmi [Hop94] oli ensimmäinen algoritmi, jonka voidaan katsoa ratkaisseen yleisen verkonmuodostusongelman järjestämättömästä pistejoukosta. Se kykenee muodostamaan verkon näytejoukosta käyttämättä apunaan muuta informaatiota kuin näytepisteiden koordinaatteja. Algoritmi osaa käsitellä pintoja, joissa on reikiä ja reunoja. Lisäksi se solveltuu myös sellaisten syötteiden käsittelyyn, joissa on kohinaa.

Hoppen edellä esiteltyä luokittelua käyttäen algoritmi kuuluu selvästi implisiittisiin pinnan uudelleenmuodostusmenetelmiin. Algoritmi toimii kahdessa osassa. Ensin muodostetaan syötejoukon perusteella etäisyysfunktio *f*, joka arvioi jonkin pisteen $x \in R^3$ etäisyyttä syötejoukon kuvaamaan pintaan. Etäisyys on etumerkillinen sen mukaan sijaitseeko piste pinnan sisä- vai ulkopuolella. Pintaan kuuluvat ne pisteet, joilla etäisyysfunktio saa arvon nolla. Kyseessä on siis implisiittistä pintaa kuvaava funktio. Menetelmän toisessa vaiheessa tästä funktiosta voidaan tuottaa verkko millä tahansa sopivalla implisiittisiä pintoja käsittelevällä algoritmilla, esimerkiksi marching cubes – algoritmilla.

Etäisyysfunktion muodostaminen on algoritmin hankalin osuus. Tehtävänä on rakentaa funktio, joka kykenee mittaamaan jonkin mielivaltaisen pisteen etäisyyttä pintaan, jota ei etukäteen tunneta. Apuna voidaan käyttää pelkästään joukkoa järjestämättömiä pisteitä. Hoppe ratkaisi ongelman käyttämällä tangenttitasoja. Jokaiselle syötejoukon pisteelle etsitään taso, joka toimii paikallisena arviona pinnan yleisestä suunnasta ja sijainnista näytepisteen kohdalla. Etäisyysfunktion arvo saadaan tällöin yksinkertaisesti mittaamalla tutkittavan pisteen etäisyys lähimmän näytepisteen tangenttitasoon. Tangenttitason etsiminen jollekin näytepisteelle käyttäen apuna sen lähimpiä naapuripisteitä onnistuu suhteellisen helposti, mutta ongelmaksi muodostuu vierekkäisten tasojen suunta. Koska etäisyysfunktio on etumerkillinen, niin tangenttitasoille pitää valita etupuoli. Kaikkien tangenttitasojen pitäisi osoittaa loogisesti samaan suuntaa joko pinnan

sisä- tai ulkopuolelle, mutta ei niin, että osa osoittaa sisäpuolelle ja osa ulkopuolelle. Tämä on varmistettava jollakin sopivalla menetelmällä. Algoritmi voisi periaatteessa käyttää myös etumerkitöntä etäisyysfunktiota, jolloin tangenttitasojen puolilla ei olisi väliä, mutta käyttämällä etumerkillistä etäisyyttä taataan, että lopputulos on monisto.

Tangenttitaso muodostetaan syötejoukon pisteelle x_i etsimällä ensin tietty määrä pisteen lähimpiä naapureita. Näiden pisteiden lukumäärä on käyttäjän antama parametri, mutta Hoppe esittää myös mahdollisen tavan, jolla sopiva arvo voitaisiin määrittää automaattisesti. Hoppe merkitsee arvoa tunnuksella *k* ja kutsuu *k*:ta pisteen lähintä naapuria pisteen *k -ympäristöksi.*. Tangenttitaso syötejoukon pisteelle x_i määritellään normaalivektorin $\overline{n_i}$ ja pisteen o_i muodostamana parina. Piste o_i saadaan selville yksinkertaisesti *k*:n lähimmän naapurin keskiarvona. Normaalivektori $\overline{n_i}$ on hieman hankalampi. Sen laskemiseksi Hoppe käytti menetelmää, joka tunnetaan nimellä *pääkomponenttianalyysi* (principal component analysis) tai *Karhunen-Loève -muunnos* (Karhunen-Loève transform). Sen muodostamista ei esitetä tarkemmin tässä yhteydessä.

Pelkkä normaalivektorin etsiminen tangenttitasolle ei siis vielä riitä, vaan sen on myös osoitettava loogisesti samaan suuntaan muiden tangenttitasojen normaalivektorien kanssa. Jos oletetaan, että näytejoukon kuvaama pinta ei sisällä teräviä kulmia ja että syötejoukko on suhteellisen tiheä ja tasaisesti jakautunut, niin vierekkäisten pisteiden tangenttitasot ovat lähes yhdensuuntaisia. Vierekäisten pisteiden tasoista on tällöin helppo havaita osoittavatko ne oikeaan suuntaan, esimerkiksi tutkimalla niiden normaalivektorien pistetuloa. Ongelmana on löytää globaali ratkaisu, jossa kaikki mahdolliset vierekkäiset parit osoittavat samaan suuntaan. Tätä ongelmaa voidaan mallintaa graafin optimoinnilla. Jokaisen tangenttitason keskipisteen ajatellaan olevan graafin solmu ja solmusta lähtee särmä kaikiin riittävän lähellä sijaitseviin keskipisteisiin. Särmään liitetään paino, joka riippuu särmän yhdistävien tasojen välisestä kulmasta. Tehtävänä on valita tangenttitasojen suunnat niin, että graafin paino maksimoituu. Ongelman voidaan osoitaa olevan NP-täydellinen, joten käytännössä joudutaan turvautumaan johonkin arvioivaan menetelmään. Hoppe ratkaisi ongelman yksinkertaisesti valitsemalla satunnaisesti tason ja levittämällä sen suunnan rekursiivisesti viereisiin tasoihin. Tässä käytettiin apuna edellä kuvatulla tavalla muodostettua graafia. Menetelmä ei kaikissa tilanteissa toimi sellaisenaan, vaan sitä täytyy muokata niin, että tason suunnan levitys tapahtuu esisijaisesti niiden tasojen välillä, joiden suunnat ovat lähellä toisiaan.

Amenta, Choi ja Kolluri [Ame98] ehdottivat vaihtoehtoisena ratkaisuna napavektorien käyttöä tangenttitasojen arvioimisessa. Napavektoreita ja niiden suhdetta pinnan nor-

maalivektoreihin käsitellään tarkemmin crust-algoritmin yhteydessä. Tämä tapa kuitenkin edellyttää, että syötejoukosta muodostetaan Voronoi-diagrammi.

Tangenttitasoja ei suoranaisesti käytetä elementteinä lopullisen mallin rakentamiseen, vaan niiden avulla vain mitataan jonkin pisteen etäisyyttää tasoon, joka näyttäisi lähiympäristön perusteella olevan sopiva paikallinen arvio etsitystä pinnasta. Menetelmän etuna on, että se soveltuu kohinaisten syötteiden käsittelyyn. Koska tangenttitaso muodostetaan eräänlaisena lähiympäristön keskiarvona, niin yksi tai muutama virheellinen näytepiste tässä ympäristössä ei vielä välttämättä aiheuta näkyvää muutosta lopputulokseen. Kohinasta johtuvat virhearvot suodattuvat pois.

Vaikka etäisyysfunktio onkin vain tasojoukon perusteella rakennettu arvio, niin se toimii samalla tavoin, kuin edellä esitetyt implisiittisiä pintoja kuvaavat funktiot. Kummatkin määrittelevät pinnan funktion nollajoukkona. Lopullisen mallin rakentamiseen etäisyysfunktiosta voidaan käyttää mitä tahansa implisiittisiä pintoja käsittelevää verkonmuodostusalgoritmia. Hoppe käytti lopullisen mallin tuottamiseen muokattua versiota marching cubes -algoritmista. Syötejoukon alue kuutioidaan käyttäjän haluamalla tarkkuudella ja marching cubes -algoritmin annetaan käsitellä ne kuutiot, joiden läpi etäisyysfunktion nollajoukko kulkee. Algoritmissa 2 esitetään Hoppen verkonmuodostusalgoritmi.

Algoritmi 2

Hoppen verkonmuodostusalgoritmi

```
Syöte: joukko pisteitä, k –parametri ja kuution koko
Tulos: kolmioverkko
    for jokaiselle syötejoukon pisteelle do
        Etsi pisteelle sen k lähintä naapuria.
        Muodosta lähimpien naapurien avulla pisteelle tangenttitaso.
    Muodosta tangenttitasojen joukosta Riemannin graafi.
    Etsi graafille pienin virittävä puu.
    Etsi puun avulla tasoille loogisesti yhtenäinen suunta.
    Jaa syötedata kuutioihin.
    Muodosta kuutioidusta datasta verkko marching cubes muunnelmalla käyttäen
    apuna tagenttitasojen joukkoa.
```

Algoritmin tehokkuus riippuu siitä, miten tietyt osaongelmat on ratkaistu. Tärkeimmät tehokkuuteen vaikuttavat osaongelmat ovat *k*:n lähimmän naapuripisteen etsiminen jollekin pisteelle, lähimmän tangenttitason origon etsiminen jollekin pisteelle ja tangenttitasojen suunnan yhdenmukaistamiseen käytetty menetelmä. Lisäksi kokonaistehokkuuteen vaikuttaa käytetty implisiittisten pintojen verkonmuodostusalgoritmi. Etsintäongelmia voi nopeuttaa käyttämällä apuna jotakin sopivaa tietorakennetta. Hoppe kuitenkin vain yksinkertaisesti jakoi pistejoukon tilan pienempiin kuutioihin ja piti kirjaa siitä mitkä pisteet ovat minkäkin kuution alueella. Tällä tavalla k:n lähimmän naapurin jollekin pisteelle voi etsiä ajassa $O(k)$. Tangenttitasojen suunnan levittämiseen käytetty Riemannin graafi voidaan muodostaa ajassa $O(nk)$. Pienimmän virittävän puun läpikäyminen on aikavaatimukseltaan luokkaa $O(n \log n)$.

Edellä mainitun *k*-arvon lisäksi Hoppen algoritmi tarvitsee käyttäjältä kaksi muuta parametriä: arviot näytetiheydestä ja kohinan määrästä syötteessä. Niitä merkitään tunnuksilla ρ ja δ. Näiden parametrien määrittäminen oikein on kriittistä algoritmin toiminnan kannalta, sillä ne vaikuttavat suoraan siihen miten pieniä yksityiskohtia näytteen perusteella muodostettuun malliin voidaan tuottaa ja mitkä kohdat pinnasta tulkitaan rei'iksi. Algoritmi olettaa, että kaksi erillistä "kerrosta" pinnasta ei koskaan ole lähempänä toisiaan kuin arvo $\rho + \delta$ (vrt. luvussa 6 esitetty poimuongelma). Parametrien tarkka määrittäminen vaatii joko tietoa syötteen tuottaneesta menetelmästä tai sitten ne täytyy määrittää kokeellisesti jokaiselle syötteelle erikseen. Algoritmin herkkyys näiden parametrien arvoille voidaan laskea sen erääksi heikkoudeksi.

Tässä esitelty verkonmuodostusalgoritmi on suunniteltu toimimaan yhdessä myöhemmin esiteltävän Hoppen optimointialgoritmin kanssa. Hoppen oman määrittelyn mukaan verkonmuodostusalgoritmin antama lopputulos on vain verkonmuodostuksen ensimmäinen vaihe, jota myöhemmissä vaiheissa voidaan parantaa optimoimalla [Hop94].

3.4 Crust-algoritmi

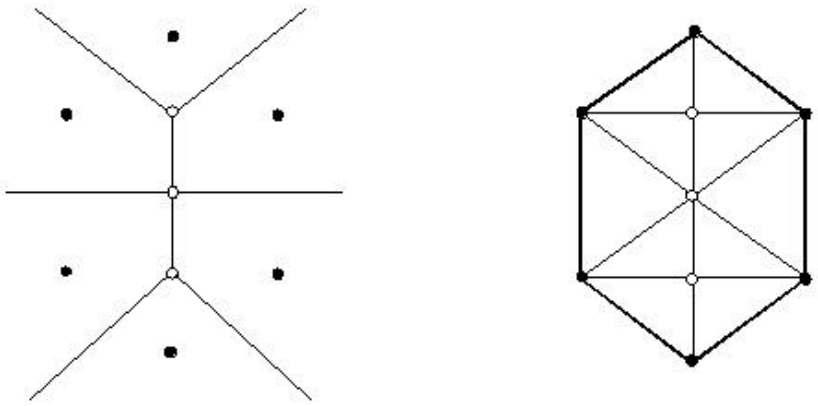
Amenta, Bern ja Kamvysselis julkaisivat vuonna 1998 menetelmän, joka tunnetaan nimellä *crust*-algoritmi [Ame98]. Se perustuu Voronoi-diagramin ja Delaunay-kolmioinnin hyödyntämiseen. Tekijät kutsuvat lopputuloksena saatavaa mallia pistejoukon kuoreksi, josta algoritmi on saanut nimensä (crust = kuori). Crust-algoritmi on ensimmäinen yleinen verkonmuodostusalgoritmi, jonka syötteelle asetetut vaatimukset osataan tarkasti määritellä [Ame01]. Aikaisemmin syötteeltä vaadittuja ominaisuuksia oli kuvailtu hyvin epämääräisin käsittein, kuten "riittävän tiheä". Amenta, Bern ja

Kamvysselis määrittelivät tarkasti mitattavat kriteerit syötteelle, jolle voidaan taata tietynlainen hyvä lopputulos.

Crust-algoritmin käsittely onkin syytä aloittaa määrittelemällä hyvä syöte. Algoritmin eräs heikkous on, että se antaa järkevän lopputuloksen vain tietynlaisille syötejoukoille. Syötejoukon pisteiden tulisi olla suhteellisen tasaisesti jakautuneita ja näytetiheyden oletetaan olevan korkeampi sellaisilla alueilla, joissa alkuperäisessä pinnassa on eniten yksityiskohtia. Käytännössä todelliset näytteet ovat tällaisia vain harvoin, joten ei ole mitään teoreettisia takeita siitä, että crust-algoritmi antaisi järkeviä lopputuloksia todellisen datan kanssa. Tekijöiden mukaan algoritmi toimii yleensä hyvin vaikka syötejoukko ei täysin täyttäisikään teoreettisia vaatimuksia ja on siten käyttökelpoinen myös käytännössä.

Crust-algoritmin kannalta hyvä syöte on sellainen, jossa näytetiheys on kääntäen verrannollinen pinnan etäisyyteen kappaleen keskiakselista. Pistejoukon S sanotaan olevan *r-näyte* pinnasta F, jos pisteen $p \in F$ Euklidinen etäisyys lähimpään näytepisteeseen on enintään r kertaa pisteen p etäisyys pinnan F keskiakseliin. Käytännössä r:n arvo 0.5 (tai sitä pienempi) riittää järkeviin lopputuloksiin. Toisin sanoen crust-algoritmi on käyttökelpoinen jos etäisyys jokaisesta pinnan pisteestä lähimpään näytepisteeseen on puolet pienempi kuin etäisyys kappaleen keskiakseliin. Erityisen ongelman muodostavat terävät kulmat ja reunat, joissa käy helposti niin, että kappaleen keskiakseli koskettaa sen pintaa. Edellä esitetyn määritelmän perusteella näytetiheyden tulisi tällöin olla äärettömän suuri järkevän lopputuloksen takaamiseksi, joten algoritmin tulee kyetä tunnistamaan tällaiset tilanteet ja käsitellä ne erikoistapauksena.

Algoritmin idea perustuu havaintoon, että mikäli syötejoukon r -arvo on rittävän pieni, niin suurin osa joukon Voronoi-kärjistä sijaitsee hyvin lähellä kappaleen keskiakselia. Itseasiassa kaksiulotteisessa tilanteessa kaikki Voronoi-kärjet ovat keskiakselin lähellä. Algoritmin perusidea onkin helpointa havainnollistaa kaksiulotteisen datan kanssa. Kuvassa 3.2 on esitetty kaksiulotteinen pistejoukko, sen Voronoi-diagrammi ja lopputuloksena saatava kuori. Kaksiulotteinen kuori rakentuu särmistä ja se määritellään seuraavasti: särmä kuuluu kuoreen mikäli pienimmän sen päätepisteiden kautta piirretyn ympyrän sisällä ei ole syötejoukon pisteitä eikä yhtäkään Voronoi-kärkeä. Mikäli syötejoukon r-arvo on pienempi kuin 0.25, niin voidaan olla varmoja siitä, että kuori yhdistää vain syötejoukon vierekkäisiä pisteitä. Syötteellä, jonka r-arvo on suurempi kuin 0.25, kuori kuitenkin yleensä edelleen yhdistää vain vierekkäisiä pisteitä, mutta siitä ei ole teoreettisiä takeita [Amb98].



Kuva 3.2 Kaksiulotteinen Voronoi-suodatus. Vasemmalla on pistejoukko S ja sen Voronoi Diagrammi. Voronoi-kärjet (tyhjät pallot) muodostavat joukon P. Oikealla on joukon $S \cup P$ Delaunay-kolmiointi. Kuoreen kuuluvat särmät on piirretty paksunnettuna.

Edellä annettu määritelmä mahdollistaa kuoreen kuuluvien särmien tunnistamisen Voronoi-kärkien avulla. Jäljelle jäävä ongelma on näiden särmien etsiminen tehokkaasti kaikkien mahdollisten pistejoukon särmien joukosta. Luvussa 2 mainittiin Delaunay-kolmioinnin erääksi ominaisuudeksi se, että sen kolmioiden kärkien kautta piirettyjen ympyröiden sisälle ei koskaan jää muita joukon pisteitä, kuin kolmion kärjet. Tätä ominaisuutta voidaan hyödyntää etsittäessä kuoreen kuuluvia särmiä. Muodostetaan ensin pistejoukko, johon kuuluvat sekä alkuperäisen syötejoukon pisteet, että sen Voronoi-kärjet. Sitten muodostetaan Delaunay-kolmiointi tästä joukosta. Kuoreen kuuluvat särmät saadaan valitsemalla kolmioinnista ne särmät, joiden kärkinä on vain alkuperäisen syötejoukon pisteitä. Tätä algoritmia kutsutaan *Voronoi-suodatukseksi* (Voronoi filtering). Kaksiulotteinen kuori on mahdollista etsiä tehokkaasti myös ilman Delaunay-kolmiointia suoraan Voronoi-diagrammista [Gol00].

Valitettavasti kolmiulotteisen datan kanssa edellä esitetty Voronoi-suodatus ei toimi sellaisenaan. Kolmiulotteisessa tilanteessa kaikki Voronoi-kärjet eivät välttämättä ole kappaleen keskiakselin lähellä, vaan näytetiheydestä riippumatta osa niistä voi olla lähempänä kappaleen pintaa. Tästä huolimatta suurin osa Voronoi-kärjistä kuitenkin seuraa edelleen kappaleen keskiakselia. Jos pystytään selvittämään mitkä Voronoi-kärjistä ovat sopivia, niin Voronoi-suodatusta voidaan käyttää myös kolmiulotteisen datan kanssa. Algoritmin kehittäjät osoittivat [Ame99], että jokaisella Voronoi-solulla on aina

sellaisia kärkiä, jotka ovat keskiakselin lähellä. Tasaisesti jakautuneessa syötejoukossa Voronoi-solut ovat muodoltaan pitkiä ja ohuita sekä melko tarkasti kohtisuoraan kappaleen pintaa vasten. Tämän seurauksena voidaan jokaisesta Voronoi-solusta valita suodatukseen mukaan ainakin ne kaksi kärkeä, jotka ovat kauimpana näytepisteestä ja toisistaan. Näitä kahta Voronoi-kärkeä kutsutaan *navoiksi*, ja niitä merkitään symboleilla p^+ ja p^-. Luokittelu positiivisiin ja negatiivisiin napoihin johtuu siitä, että silloin kun Voronoi-solu on edellä kuvatulla tavalla riittävän pitkä ja kapea, niin p^+ ja p^- ovat kappaleen pinnan vastakkaisilla puolilla. Käyttämällä Voronoi-suodatuksessa kaikkien Voronoi-kärkien sijasta vain napoja, saadaan algoritmi toimimaan myös kolmannessa ulottuvuudessa. Algoritmissa 3 esitetään perusversio crust-algoritmista.

Algoritmi 3

Crust-algoritmi

Syöte: joukko pisteitä

Tulos: kolmioverkko

Muodosta tyhjä pistejoukko P.

Muodosta syötejoukon Voronoi-diagrammi.

For jokaiselle syötejoukon pisteelle **do**

Etsi pisteen Voronoi-solu.

Etsi solun kauimmainen Voronoi-kärki p^+ (positiivinen napa).

Etsi sille vastakkainen napa p^-.

Lisää navat ja piste joukkoon P.

Muodosta Delaunay-kolmiointi joukolle P.

Etsi ne kolmiot, joissa on vain alkuperäisen syötejoukon pisteitä kärkinä ja muodosta niistä verkko.

Syötteen laadusta ja tiheydestä riippumatta edellä esitetty algoritmi saattaa tuottaa hyvin ohuita kolmioita, jotka ovat kohtisuoraan pintaan nähden. Tämän takia tarvitaan vielä ylimääräinen suodatusvaihe, jossa mahdolliset ei-toivotut kolmiot karsitaan pois. Algoritmin kehittäjät ovat osoittaneet [Ame99], että verkon kärjestä s kärjen napaan p^+ muodostettu vektori sp^+ on aina lähes ortogonaalinen pintaan nähden, eli siis hyvä arvio pinnan normaalista pisteen s kohdalla. Arvion tarkkuus riippuu syötteen r-arvosta. Tämän perusteella voidaan karsia pois ne kolmiot, joiden normaalivektori eroaa riittävästi vektoreista sp^+ tai sp^-. Vaihetta kutsutaan normaalisuodatukseksi ja se takaa,

että verkon kolmioiden normaalivektorit lähestyvät pinnan normaalivektoreita näytetiheyden kasvaessa.

Crust-algoritmin tuottama kuori ei välttämättä ole vielä sellaisenaan haluttu lopputulos, eli paloittain lineaarinen, topologisesti yhtenevä vastine alkuperäiselle pinnalle. Kuori saattaa kuitenkin sisältää tällaisen pinnan, mutta vielä normaalisuodatettukin kuori voi sisältää kolmioita, jotka eivät siihen kuulu. Niiden poistaminen riippuu siitä, millainen alkuperäinen pinta on. Jos alkuperäisessä pinnassa on reunoja tai teräviä kulmia, niin tällä hetkellä ei tunneta menetelmää, jolla ylimääräiset kolmiot voitaisiin varmuudella poistaa. Tässä tapauksessa lopulliseksi malliksi on pakko hyväksyä koko kuori. Samoin käy, jos alkuperäistä pintaa ei tunneta, jolloin on pakko olettaa sen sisältävän reunoja tai teräviä kulmia. Mutta jos alkuperäisen pinnan tiedetään olevan reunaton ja tasainen, niin kuoresta voidaan etsiä pinnan paloittain lineaarinen, topologisesti yhtenevä vastine. Ylimääräiset kolmiot karsitaan tällöin terävien kulmien avulla. Terävällä kulmalla tarkoitetaan sellaista särmää, johon liityviä kolmioita on vain toisella puolella jotakin särmän kautta kulkevaa tasoa ja joka on suurinpiirtein kohtisuora pintaan nähden. Tällaiseen särmään liityviä kolmioita ei voi olla halutussa lopputuloksessa ja ne voidaan poistaa. Lopullinen malli saadaan ottamalla jäljelle jääneiden kolmioiden ulkopuolet. Crust-algoritmi eroaa muista tässä työssä käsitellyistä menetelmistä siinä, että sen tuottaman verkon kärkinä ovat vain ja ainoastaan alkuperäisen syötejoukon pisteet.

Crust-algoritmin tehokkuus riippuu Voronoi-diagrammin ja Delaunay-kolmioinnin muodostamiseen käytetyistä menetelmistä. Kaikki muut algoritmin vaiheet ovat aikavaatimukseltaan lineaarisia. Pahimmassa tapauksessa kolmiulotteisen Delaunay-kolmioinnin aikavaatimus on luokkaa $O(n^2)$, joka on siten myös koko crust-algoritmin aikavaatimus. Käytännössä pahinta tapausta ei kuitenkaan juuri esiinny.

3.5 Power crust -algoritmi

Crust-algoritmin jälkeen Amenta tutki aihetta edelleen Choin ja Kollurin kanssa. Lopputuloksena syntyi power crust-algoritmina [Ame01] tunnettu menetelmä. Periaatteeltaan Power crust –algoritmi on edelleen kehitetty versio crust-algoritmista. Myös se perustuu samaan keskiakselin ja Voronoi-diagrammin välisen suhteen hyödyntämiseen. Power crust –algoritmi rakentaa kuitenkin lopullisen verkon täysin crust-algoritmista poikkeavalla tavalla, joten niitä voidaan perustellusti pitää eri algoritmeina. Power crust -algoritmi kykenee selviytymään myös sellaisista syötteistä, joilla crust-algoritmi ei an-

na järkevää lopputulosta. Itseasiassa tekijät ovat osoittaneet että algoritmi palauttaa "vesitiiviin" pinnan mistä tahansa syötejoukosta.

Power crust -algoritmi perustuu toisessa luvussa esitellyn keksiakselimuunnoksen hyödyntämiseen ja havaintoon siitä, että keskiakselimuunnoksen arvio saadaan kätevästi muodostettua crust-algoritmin yhteydessä kuvattujen Voronoi-napojen avulla. Tätä varten power crust -algoritmi muostaa ensiksi syötejoukon Voronoi-diagrammin ja etsii Voronoi-solujen navat. Crust-algoritmin esittelyn yhteydessä mainittiin, että navat sijaitsevat lähellä kappaleen keskiakselia, joten ne sopivat sellaisenaan keskiakselimuunnoksen maksimaalisten tyhjien pallojen keskipisteiksi. Pallon säteeksi asetetaan etäisyys navasta lähimpään syötepisteeseen. Algoritmin kehittäjät kutsuvat tällaista palloa *napapalloksi* (polar ball). Sopiva keskiakselimuunnoksen arvio saadaan muodostamalla kaikkien napapallojen joukko. Koska kyseessä on vain arvio, niin aidosta keskiakselimuunnoksesta poiketen osa napapalloista saattaa leikata kappaleen pintaa, mutta vain hyvin pienellä marginaalilla jonka suuruus riippuu syötejoukon r –arvosta.

Keskiakselimuunnoksen maksimaalisten tyhjien pallojen tavoin osa napapalloista sijaitsee kappaleen sisäpuolellla ja osa ulkopuolella. Sisäpuoliset pallot leikkaavat vierekkäisiä sisäpuolisia palloja ja ulkopuoliset pallot vastaavasti vierekkäisiä ulkopuolisia palloja, mutta koskaan sisä- ja ulkopuolinen pallo eivät leikkaa toisiaan paitsi hyvin pienellä, arvion epätarkkuudesta johtuvalla marginaalilla. Tätä tietoa voidaan käyttää hyväksi kappaleen pinnan etsimisessä. Etsitään kaksi vierekkäistä palloa, joista toinen on sisäpallo ja toinen ulkopallo. Keskiakselimuunnoksen määritelmän perusteella tiedetään nyt, että kappaleen pinnan on kuljettava näiden pallojen välistä. Tarkemmin sanottuna pinta leikkaa pallojen keskipisteiden välisen janan kohdassa, jonka määrää pallojen säteiden suhde. Ongelmaksi muodostuu miten pallot luokitellaan sisä- ja ulkopalloihin, sekä miten tästä luokittelusta rakennetaan kolmiulotteinen malli. Kummankin ongelman voi ratkaista käyttämällä erityistä painotettua versiota Voronoi-diagrammista, josta käytetään englanninkielistä termiä *power diagram*. Käytän siitä tässä työssä suomenkielistä termiä *painotettu Voronoi-diagrammi*.

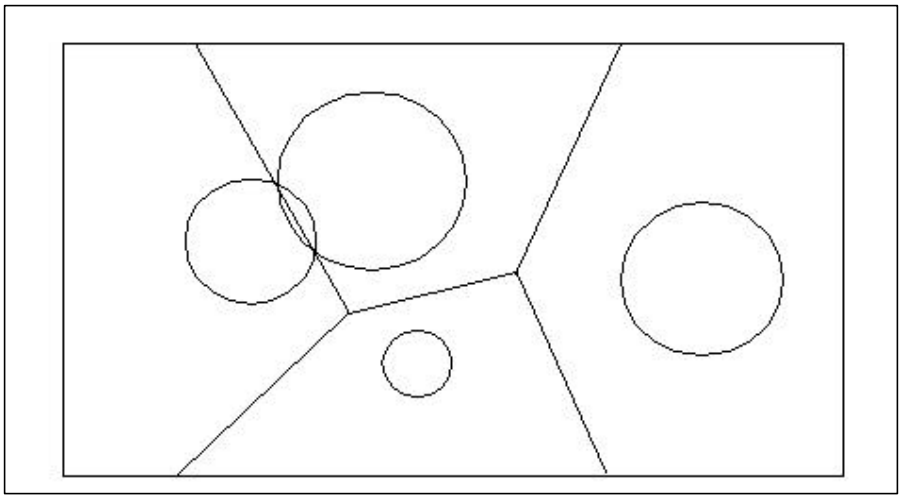
Painotettua Voronoi-diagrammia varten tarvitaan käsite, josta Amenta, Choi ja Kolluri käyttävät englanninkielistä termiä *power distance*. Koska termi on monimerkityksinen ja sitä hyvin vaikea kääntää lyhyesti, käytän siitä suomenkielistä termiä *painotettu etäisyys*. Oletetaan, että on olemassa pallo $B_{c,r}$, jolla on keskipiste c ja säde r. Tällainen pallo voidaan mieltää painotetuksi pisteeksi, siten, että pallon säde ilmaisee pisteen

painon. Painotettu etäisyys kertoo tavallisen pisteen x ja painotetun pisteen P etäisyyden toisistaan ja se määritellään seuraavasti

$$d_{pow}\left(x, P_{c,r}\right) = d^2\left(x, c\right) - r^2, \tag{2.1}$$

missä funktio d on tavallinen Euklidinen etäisyys kahden pisteen välillä. Eli d_{pow} on negatiivinen, kun piste on pallon sisällä ja positiivinen, kun piste on pallon ulkopuolella.

Painotettu Voronoi-diagrammi jakaa tilan soluihin samalla tavalla kuin tavallinen Voronoi-diagrammikin, mutta solun muodostumisen kriteerinä ei käytetä tavallista etäisyyttä vaan painotettua etäisyyttä. Solun seinämä ei nyt leikkaa kahden pisteen välistä janaa sen puolessa välissä, kuten tavallisessa Voronoi-diagrammissa, vaan painokertoimien suhteen määräämässä kohdassa. Ja koska painokerroin on sidottu pallon säteeseen, niin sisäpallon ja ulkopallon välille muodostuneen solun seinä kulkee sopivasti juuri kappaleen pinnan kohdalta tai muunnoksen tarkkuudesta riippuen ainakin hyvin läheltä sitä. Voronoi-diagrammin tavoin myös painotetun Voronoi-diagrammin solut ovat muodoltaan konvekseja, joten hyvä paloittain lineaarinen arvio kappaleen pinnasta saadaan valitsemalla painotetusta Voronoi-diagrammista kaikki sisä- ja ulkosolujen väliset seinämät. Tätä rajakerrosta sisä- ja ulkosolujen välillä algoritmin kehittäjät kutsuvat nimellä *power crust* ja se on algoritmin antama lopputulos. Painotetun Voronoi-diagrammin modostamiseen voidaan käyttää samoja menetelmiä kuin tavallisen Voronoi-diagramminkin. Esimerkki kaksiulotteisesta painotetusta Voronoi-diagrammista on esitetty kuvassa 3.3. Kuvassa on neljän painotetun pisteen joukko (ympyrät) ja niille muodostetun painotetun Voronoi-diagrammin solut.



Kuva 3.3 Kaksiulotteinen painotettu Voronoi-diagrammi

Nyt täytyy vielä kyetä luokittelemaan painotetun Voronoi-diagrammin solut sisä- ja ulkopuolisiin soluihin, jotta haluttu rajakerros voidaan etsiä. Painotetun Voronoi-diagrammin solujen luokittelu on kuitenkin olennaisesti helpompaa, kuin pelkkien mapapallojen, sillä sen soluista voidaan muodostaa graafi, jota läpikäymällä luokittelu helpottuu.

Algoritmi 4 esittää pseudo-koodi luonnoksen power crust -algoritmista. Vertaamalla sitä algoritmiin 3, on helppo havaita sukulaisuussuhde crust-algoritmin kanssa.

Algoritmi 4

Power crust –algoritmi

Syöte: joukko pisteitä

Tulos: monikulmioverkko

Muodosta tyhjä pistejoukko P.

Muodosta syötejoukon Voronoi-diagrammi.

For jokaiselle syötejoukon pisteelle **do**

Etsi pisteen Voronoi-solu.

Etsi solun kauimmainen Voronoi-kärki p^+ (positiivinen napa).

Etsi sille vasttakkainen napa p^-.

Lisää navat ja piste joukkoon P.

Muodosta painotettu Voronoi-diagrammi joukolle P.

Luokittele painotetun Voronoi-diagrammin solut sisä- ja ulkopuolen soluihin.

Etsi sisä- ja ulkopuolen väliseen rajakerrokseen kuuluvat solujen seinät ja muodosta niistä verkko.

Power crust -algoritmin täytyy käsitellä terävät kulmat erikoistapauksena. Crust-algoritmin yhteydessä todettiin, että jos syötejoukon pisteet ovat tasaisesti jakautuneita, niin Voronoi-solut ovat muodoltaan pitkiä ja kapeita, sekä lähes kohtisuoraan kappaleen pintaa vasten. Terävien kulmien kohdalla tämä sääntö ei kuitenkaan päde ja seuraus on, että pinta ei rakennu oikein muodoltaan vääristyneiden Voronoi-solujen kohdalla. Toisaalta tämä tarjoaa mahdollisuuden myös tunnistaa terävät kulmat tutkimalla Voronoi-solujen muotoa. Kulmien ongelma ratkeaa yksinkertaisesti hylkäämällä sellaiset napaparit, joissa jompikumpi Voronoi-solu ei täytä muodolle asettettuja vaatimuksia. Menetelmä on tehokas, se kykenee esimerkiksi tuottamaan oikein muodostuneen terävän särmän sellaisessa tilanteessa, jossa yksikään näytejoukon pisteistä ei sijaitse särmällä. Ongelma on se, että muodoltaan vääristyneitä Voronoi-soluja voi syntyä muutenkin kuin terävien kulmien seurauksena, joten niitä ei voi oletusarvoisesti hylätä. Käyttäjän onkin päätettävä käytetäänkö vääristyneiden solujen hylkäystä ja mitkä ovat hylätyksi tulemisen kriteerit. Tämä vaatii ennakkotietoa näytejoukon kuvaamasta kappaleesta.

Power crustin -algoritmin tuottama lopputulos ei ole kolmioverkko. Myöskään kaikki syötejoukon pisteet eivät aina ole lopputuloksena saatavan verkon kärkiä ja toisaalta taas kaikki verkon kärjet eivät välttämättä ole syötejoukon pisteitä. Jos verkko muute-

taan kolmioverkoksi, niin käytetty kolmiointimenetelmä voi vaikuttaa lopputuloksen laatuun kolmioiden muodon osalta.

Power crust -algoritmi tuottaa siis perusmuodossaan aina "vesitiiviin" pinnan mistä tahansa syötejoukosta. Tästä seuraa, että perusmuodossa algoritmilla ei voi käsitellä pintoja, joissa on reikiä. Algoritmiin on kehitetty myös sellainen laajennus, joka kykenee käsittelemään reikiä.

4. Verkon optimointimenetelmiä

Edellä esitellyt algoritmit pyrkivät toistamaan mahdollisimman tarkasti syötejoukon kuvaaman pinnan, mutta ne eivät juurikaan kiinnitä huomiota lopputuloksena syntyvän verkon laatuun. Verkko saattaa sisältää huomattavasti enemmän monikulmioita, kuin olisi tarpeen halutun visuaalisen informaation välittämiseen. Tekniikan kehittyessä kolmiulotteista kuvadataa tuottavien menetelmien tarkkuus kasvaa jatkuvasti. Esimerkiksi lääketieteellisillä instrumenteilla kyetään jo gigavokseleissa laskettavaan mittaustarkkuuteen. Kun tällaisesta tietomäärästä muodostetaan polygoniverkko jollakin edellä kuvatuista algoritmeista, se saattaa sisältää kymmeniä miljoonia kolmioita. Tämän kokoluokan verkon esittäminen nykyisillä laitteilla on hidasta ja ainakin reaaliaikaisuudesta joudutaan luopumaan. Vaikka visualisointiin tarkoitettujen laitteistojen kehitys on viimevuosina ollut nopeaa, niin tuotetun kuvadatan koko ja monimutkaisuus tuntuu silti edelleen kasvavan nopeammin kuin sen esittämiseen tarkoitetut laitteet nopeutuvat.

Suuren määrän lisäksi edellä kuvatut algoritmit saattavat tuottaa myös täysin turhia kolmioita. Tämä voi johtua paikallisesti liian korkasta näytetiheydestä. Esimerkiksi käy neliön muotoinen tasainen pinta, josta on otettu hyvin suuri määrä näytteitä. Kun näiden näytteiden pohjalta muodostetaan malli se sisältää suuren määrän kolmioita, vaikka neliön kuvaamiseen täydellisesti riittäisi kaksi kolmiota.

Yksi tapa etsiä ratkaisuja edellä esitettyihin ongelmiin on verkon optimointi. Verkkoa optimoitaessa otetaan lähtökohdaksi alkuperäinen verkko ja muutetaan sitä jollakin menetelmällä yksinkertaisempaan muotoon. Tarkoitus on pienentää monikulmioiden lukumäärää, jolloin verkon vaatima tila- ja käsittelyaika pienenee. Toinen mahdollinen tavoite on, että optimoitu verkko kuvaisi alkuperäistä verkkoa paremmin haluttua pintaa. Yleensä nämä kaksi tavoitetta ovat ristiriidassa, sillä monikulmioiden vähentäminen pienentää verkon ilmaisuvoimaa. Tästä syystä optimointimenetelmien tulee jollakin tavalla arvioida verkon laatua optimointiprosessin aikana. Tässä luvussa esiteltävät menetelmät osoittavat, että päätös verkon muuttamisesta voi perustua joko koko verkon laatua kuvaavaan arvioon tai pelkästään paikalliseen, parhaillaan optimoitavan kohdan ympäristöstä muodostettuun arvioon. Perimmäinen tavoite optimointialgoritmeilla on löytää sopiva tasapaino verkon monimutkaisuuden ja kuvauksen laadun välillä.

Hoppe [Hop94] muotoilee verkon optimointiongelman seuraavasti: oletetaan, että on olemassa joukko datapisteitä ja jokin verkko M_0, joka jollakin tarkkuudella vastaa datapisteiden kuvaamaa pintaa. Tavoite on löytää verkko M, joka on samaa topologista tyyppiä verkon M_0 kanssa ja joka sekä vastaa hyvin datapisteiden kuvaamaa pintaa, että sisältää mahdollisimman vähän kärkiä. Hoppen muotoilu on mielenkiintoinen, koska se olettaa, että optimointialgoritmilla käytössään joukko datapisteitä, jonka perusteella kuvauksen laatu voidaan määrittää. Tämän perusteella optimointialgoritmit voidaankin luokitella kahteen ryhmään:

1. Algoritmit, jotka optimoivat verkkoa puhtaasti geometrisin perustein tuntematta verkon kuvaamaa alkuperäistä pintaa.

2. Algoritmit, jotka vaativat jotakin tietoa verkon kuvaamasta pinnasta, kyetäkseen tuottamaan optimaalisen verkon. Tieto voi olla esimerkiksi se näytepisteiden joukko, jonka perusteella verkon ensimmäinen versio on muodostettu.

Toinen tapa luokitella verkon optimointiin tarkoitetut algoritmit on jakaa ne kahteen ryhmään sen perusteella suoritetaanko optimointi kokonaan etukäteen, vai voidaanko optimoinnin määrään vaikuttaa vielä myöhemmin verkkoa piirrettäessä.

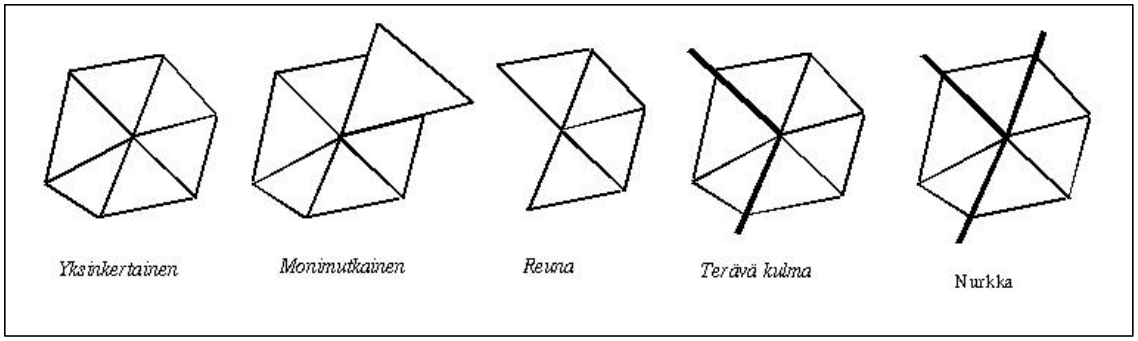
1. *Staattiset algoritmit* saavat syötteenä alkuperäisen verkon ja palauttavat lopputuloksena optimoidun verkon. Ne ajetaan käsiteltävälle datalle kerran, jonka jälkeen optimoitu verkko on lopullinen.

2. *Dynaamiset algoritmit* pyrkivät tuottamaan syötteenä annetusta verkosta tietorakenteen, jonka avulla verkko voidaan esittää usealla eri tarkkuudella ja haluttu tarkkuus voidaan valita ajoaikaisesti. Esimerkiksi katsojasta hyvin kaukana olevat kohteet voidaan esittää pienemmällä tarkkuudella kuin lähellä olevat, koska katsoja ei kuitenkaan erottaisi kaukaisista kohteista kaikkia yksityiskohtia. Eräät näistä algoritmeista (esimerkiksi [Hop97c]) mahdollistavat myös sen, että datasta voidaan valita jokin osa tai alue, joka esitetään tarkasti ja kiinnostuksen kohteen ulkopuolella olevat osat pienemmällä tarkkuudella. Tässä työssä käsitellään lähinnä staattiseen luokkaan kuuluvia algoritmeja, mutta energiafunktiomenetelmien yhteydessä esiteltyjen operaatioiden avulla on mahdollista toteuttaa myös dynaamisesti ajoaikana optimoituvia verkkoja [Hop97b].

4.1 Verkon harventaminen

Harvennus nimikkeen alle voidaan laittaa lukuisia verkon muokkaukseen tarkoitettuja menetelmiä, mutta tässä esitellään Schroederin, Zargen ja Lorensenin [Sch97] kehittämä menetelmä, joka on eräs yksinkertaisimmista tavoista optimoida verkkoa. Algoritmi käy läpi verkon kärjet ja tutkii pinnan paikallista muotoa kärjen ympäristössä. Jos ympäristö täyttää tietyt kriteerit, niin kärki sekä siihen liittyvät kolmiot poistetaan verkosta. Syntynyt reikä paikataan ja lopputuloksena on vähemmän kolmioita, kuin alkuperäisessä verkossa. Algoritmi saattaa käydä suorituksen aikana verkon kärjet useaan kertaan läpi, yleensä niin kauan kunnes poistokriteerit täyttäviä kärkiä ei enää löydy.

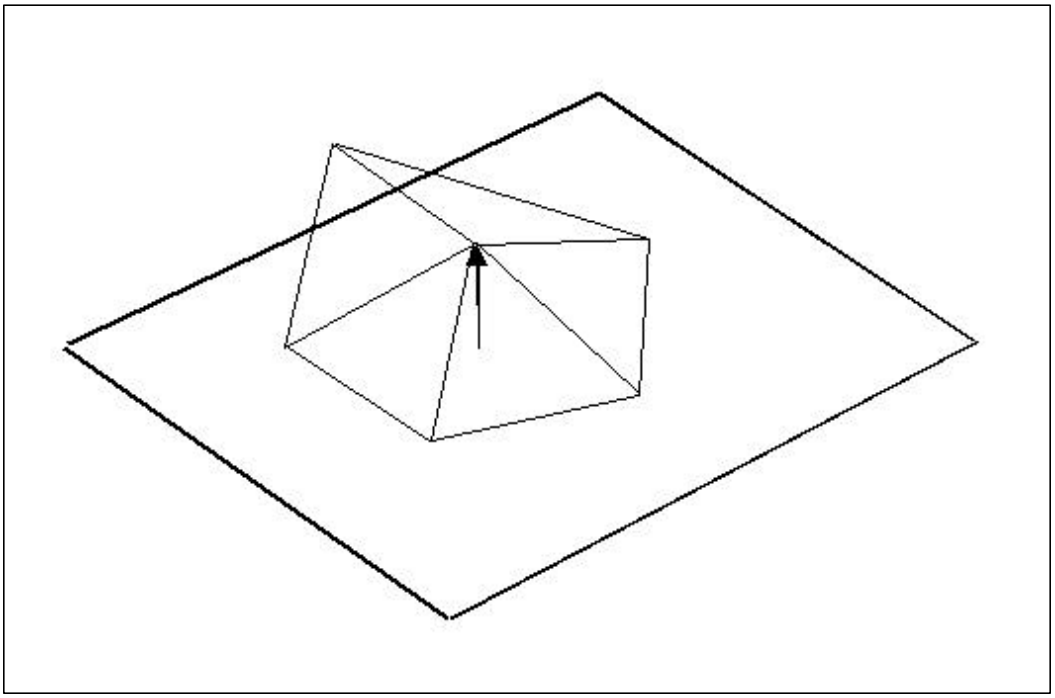
Kärjen ympäristöllä tarkoitetaan kaikkien niiden kolmioiden joukkoa, joihin valittu kärki kuuluu. Ympäristöön kuuluvat kolmiot muodostavat valitun kärjen ympärille alueen, jota tekijät kutsuvat silmukaksi. Silmukka vastaa luvussa 2 esitettyä Hoppen määritelmää kärjen ympäristöstä. Erilaisia silmukoita voi olla viittä eri tyyppiä ja ne on esitetty kuvassa 4.1. *Yksinkertainen* silmukka on tilanne, jossa jokainen kärjestä lähtevä särmä kuuluu täsmälleen kahteen kolmioon. Jos tämä ehto ei täyty, niin silmukka on tyyppiä *monimutkainen* (tällöin verkko ei ole monisto). Yksinkertainen silmukka voi lisäksi olla tyyppiä *reuna, terävä kulma* tai *nurkka*. Reuna on tilanne, jossa jokin kärjestä lähtevistä särmistä kuuluu pinnassa olevaan reikään tai reunaan. Terävä kulma ja nurkka –tyypit vaativat hieman lisäselvitystä. Jos kahden vierekkäisen, silmukkaan kuuluvan kolmion välinen kulma ylittää tietyn arvon, niin tällöin silmukka käsitellään erikoistapauksena. Tämän kulman suuruus on käyttäjän antama parametri. Jos silmukassa on kaksi tällaista kulmaa, niin silloin se on tyyppiä *terävä kulma*. Jos taas silmukassa on vain yksi tai yli kaksi kynnysarvon ylittävää kulmaa, niin se on tyyppiä *nurkka*.



Kuva 4.1 Mahdolliset silmukkatyypit

Kaikki muut, paitsi luokkaan *monimutkainen* ja luokkaan *nurkka* kuuluvat silmukat ovat mahdollisia kandinaateja kärjen poistoa ajatellen ja seuraavaksi tutkitaan täyttääkö

silmukka poistolle asetetut kriteerit. Sopivia kriteerejä on olemassa useita. Voidaan esimerkiksi tutkia kärjen etäisyyttä silmukan reunoihin tai sen etäisyyttä niin sanottuun keskimääräiseen tasoon. Schroeder [Sch97] käytti keskimääräisen tason menetelmää. Siinä kärjelle muodostetaan sen ympäristön suuntaa ja sijaintia kuvaava keskimääräinen taso. Keskimääräinen taso vastaa Hoppen verkonmuodostusalgoritmin tangenttitasoa, mutta sen muodostaminen on huomattavasti yksinkertaisempaa, koska käytössä on jo pisteeseen liittyvät kolmiot ja niiden normaalivektorit. Jos kärjen etäisyys keskimääräisestä tasosta on pienempi kuin tietty kynnysarvo, niin se voidaan poistaa. Tässä muodossa algoritmi tutkii onko kärki sellainen, että verkon pinta muuttuu sen kohdalla merkittävästi lähiympäristöön verrattuna. Jos pinta on kärjen kohdalla yleisesti hyvin tasainen, niin kärki voidaan poistaa ilman että siitä aiheutuu suurta näkyvää muutosta. Poistamisen laukaiseva kynnysarvo on käyttäjän määrittelemä parametri, jolla voidaan suoraan vaikuttaa lopputuloksena saatavan verkon laatuun ja kolmioiden lukumäärään. Silmukka, keskimääräinen taso ja kärjen etäisyys tasosta on esitetty kuvassa 4.2.



Kuva 4.2 Kärjen etäisyys keskimääräisestä tasosta

Kärjen lisäksi verkosta poistetaan myös sen ympäristöön kuuluvat kolmiot. Tällöin verkkoon syntyy reikä, joka täytyy paikata jollakin sopivalla kolmiointimenetelmällä. Schoeder esitti yksinkertaisen rekursiivisen algoritmin, joka soveltuu tähän tapaukseen. Koska syntyneen reiän reunapisteille on jo laskettu keskimääräinen taso, niin sitä voidaan hyödyntää myös aukon kolmioinnissa. Algoritmi toimii seuraavasti: valitaan kaksi pistettä, jotka eivä ole vierekkäin silmukassa ja muodostetaan niiden välille särmä.

Muodostetaan sitten taso, joka kulkee saadun särmän kautta ja on kohtisuorassa keskimääräistä tasoa vastaan. Sitten jaetaan silmukan pisteet kahteen joukkoon sen perusteella ovatko ne tason etu- vai takapuolella (tasolla olevat pisteet kuuluvat kumpaankin joukkoon). Sitten suoritetaan kummallekin uudelle joukolle rekursiivisesti samat toimenpiteet, kunnes jäljellä on kolme pistettä. Niistä muodostetaan kolmio, joka lisätään verkkoon. Edellä esitetty menetelmä voi tuottaa samasta silmukasta useita erilaisia kolmiointeja riippuen siitä miten jakavat tasot valitaan, joten tason valinnassa kannattaa käyttää sopivia kriteerejä parhaan lopputuloksen saavuttamiseksi. Algoritmi 5 havainnollistaa harvennusalgoritmin Lorensenin, Schroederin ja Zargen esittämässä muodossa

Algoritmi 5

Lorensenin, Schroederin ja Zargen harvennusalgoritmi

```
Syöte: verkko
Tulos: optimoitu verkko
    For jokaiselle verkon kärjelle do
        Muodosta kärjelle silmukka.
        Selvitä silmukan tyyppi.
        If (silmukan tyyppi ei ole "monimutkainen" and silmukan tyyppi ei ole "nurkka") do
            Etsi silmukalle keskimääräinen taso.
            If (kärjen etäisyys keskimääräiseen tasoon on pienempi kuin kynnysarvo) do
                Poista kärki ja silmukkaan kuuluvat kolmiot verkosta.
                Kolmioi syntynyt reikä umpee jollakin sopivalla menetelmällä.
```

4.2 Optimointi energiafunktiota minimoimalla

Energiafunktioihin perustuvissa menetelmissä verkolle määritellään energia, jonka määrä riippuu siitä millainen verkko on. Energia muodostuu kahdesta tekijästä: verkon monimutkaisuudesta ja kuvauksen laadusta. Mitä monimutkaisempi verkko on, sitä suurempi on sen energia. Kuvauksen laadulla tarkoitetaan sitä, miten hyvin verkko kuvaa pintaa, jonka perusteella se on mallinnettu. Mitä huonompi laatu on, sen suurempi on verkon energia. Energia on siis minimissään silloin kun verkko on mahdollisimman yksinkertainen, mutta kuvaa mahdollisimman hyvin haluttua pintaa. Verkon energiaa voi-

daan pienentää yksinkertaistamalla verkkoa, mutta tämä täytyy suorittaa siten että kuvauksen laatu ei heikkene, koska laadun heikkeneminen kasvattaisi energian takaisin ennalleen tai jopa sen yli. Energiafunktiomenetelmien perusideana on muodostaa verkon energiaa kuvaava funktio, jota yritetään minimoida muuttamalla verkkoa tietyillä operaatioilla.

Energiafunktiomenetelmät perustuvat Hoppen jo edellä mainittussa väitöskirjassaan esittelemään menetelmään [Hop94]. Tämä menetelmä esiteltiin tarkemmin Hoppen, DeRosen ja Duchampin julkaisemassa tutkimuksessa [Hop97a], jossa Hoppen alkuperäistä ideaa on viety eteenpäin. Lisäksi Hoppe kehitti menetelmää vielä edelleen tutkiessaan dynaamisesti optimoituvia verkkoja [Hop97b].

Energiafunktio voidaan määritellä ja painottaa monella tavalla riippuen siitä, mitä verkon ominaisuuksia pidetään tärkeinä. Esittelen tässä malliksi energiafunktion, jonka Hoppe esitti alunperin väitöskirjassaan. Selvitän miten funktio muodostuu ja miten verkon ominaisuudet vaikuttavat sen arvoon. Hoppe määrittelee energiafunktion verkolle (K,V) seuraavasti

$$E(K,V) = E_{dist}(K,V) + E_{rep}(K) + E_{spring}(K,V). \qquad (3.1)$$

Termit E_{dist} ja E_{rep} vastaavat verkon optimoinnin kahta päätavoitetta, joiden mukaan verkon tulisi ensinäkin vastata mahdollisimman hyvin alkuperäisen syötejoukon kuvaamaa pintaa ja toisekseen sen tulisi sisältää mahdollisimman vähän kärkiä. Ensimmäisen tavoitteen saavuttamiseksi Hoppen optimointialgoritmi tarvitsee alkuperäistä syötejoukkoa, jonka perusteella malli on rakennettu. Ensimmäinen termi E_{dist} muodostetaan laskemalla yhteen syötejoukon $X = \{x_1,...,x_n\}$ jokaisen pisteen neliöllinen etäisyys sen hetkiseen verkkoon. Termi määritellään seuraavasti

$$E_{dist}(K,V) = \sum_{i=1}^{n} d^2(x_i, f_v(|K|)). \qquad (3.2)$$

Kaavassa funktio d^2 ilmoittaa kahden kohteen välisen lyhimmän neliöidyn etäisyyden. Muuttuja $f_v(|K|)$ tarkoittaa verkon (K,V) *geometrista realisaatiota*, eli yksinkertaisesti verkkoa, joka saadaan liittämällä pistejoukko V simplekseistä rakennettuun systeemiin *K*. Termin E_{dist} avulla saadaan aikaan verkkoja, joiden pinnat kulkevat mahdollisimman läheltä syötejoukon pisteitä, vaikka syötejoukon pisteet muuten eivät olisikaan verkon kärkiä.

Toinen termi E_{rep} vaikuttaa kärkien lukumäärään. Jos verkkoon lisätään kärki, niin etäisyysenergia E_{dist} todennäköisesti pienenee. Koska tavoitteena on verkko, jossa on mahdollisimman vähän kärkiä, niin tarvitaan termi, joka kasvattaa sopivasti energiaa, kun kärkiä lisätään. Termi E_{rep} määritellään seuraavasti

$$E_{rep}(K) = c_{rep} m . \tag{3.3}$$

Kaavassa *m* on verkon kärkien lukumäärä ja c_{rep} on käyttäjän määrittelemä parametri, jolla voidaan säätää tasapainoa verkon monimutkaisuuden ja geometrisen laadun välillä.

Kolmas termi lisättiin sen takia, että energiafunktiolle voitaisiin taata sopivan minimin olemassaolo. Pelkästään termeistä $E_{dist} + E_{rep}$ muodostetulle funktiolle ei aina ole olemassa haluttua minimiä. Yritettäessä optimoida verkkoa tällaisessa tapauksessa saattaa lopputulokseen syntyä ikäviä häiriöitä. Verkon lähiympäristöön syntyy ylimääräisiä haamukärkiä, jotka muodostavat malliin epänormaaleja, teräviä piikkejä. Termillä E_{spring} lisätään jokaiseen verkon särmään jousi, jonka lepopituus on nolla. Jousi pyrkii vetämään särmän mahdollisimman lyhyeksi. Mitä pidempi särmä on, sitä enemmän verkkoon on varastoitunut energiaa. Jouselle käytetään jousivakiota *k*, joka on sama kaikille verkon särmille. Termi E_{spring} määritellään seuraavasti

$$E_{spring}(K,V) = \sum_{\{j,k\} \in K} k \left\| v_j - v_k \right\|^2 . \tag{3.4}$$

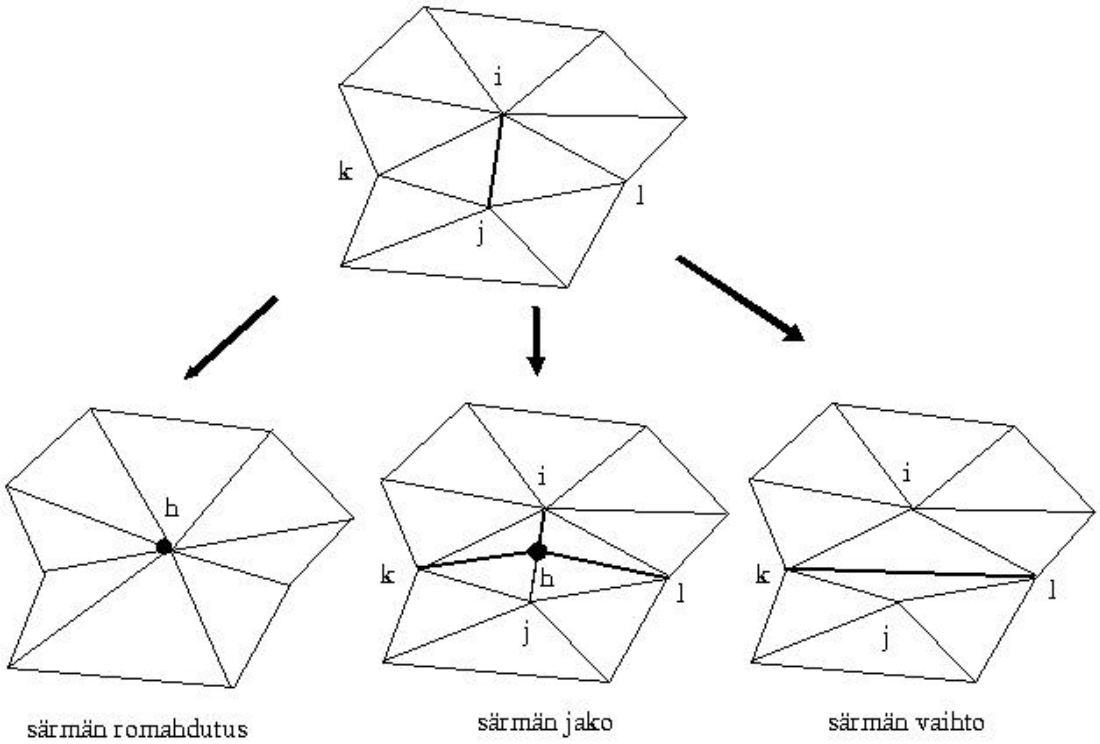
Jousitermin tarkoitus ei kuitenkaan ole eliminoida verkossa luonnostaan esiintyviä teräviä piikkejä, vaikka se yrittääkin vetää särmät mahdollisimman lyhyeksi. Sen tarkoitus on yhdessä E_{dist} ja E_{rep} termien kanssa ohjata optimointiprosessia kohti energiafunktion paikallista minimiä jonkin särmän läheisyydessä. Käytännössä parhaat tulokset saavutetaan, kun jousivakiota pienennetään pikkuhiljaa optimoinnin edetessä.

Varsinainen optimointiongelma, funktion $E(K,V)$ minimointi jonkin verkon (K,V) suhteen, voidaan ratkaista jakamalla tehtävä kahteen sisäkkäiseen osaongelmaan. Sisempi silmukka optimoi joukon *V* kärkien sijainnin sen hetkisen systeemin *K* suhteen ja ulompi silmukka optimoi systeemiä *K*.

Sisemmän silmukan tarkoitus on löytää sen hetkisen verkon kärjille sellainen sijainti, että energiafunktio on pienimmillään. Tässä vaiheessa ei lisätä tai poisteta verkon elementtejä, vaan ainoastaan siirretään kärkiä kohti pienemmän energian tilaa. Sisemmässä

silmukassa voidaan unohtaa energiafunktion termi E_{rep}, koska kärkien sijainti ei vaikuta siihen ja keskittyä optimoimaan kaavaa $E_{dist} + E_{spring}$.

Kun kärjille on ensin löydetty sisemmässä silmukassa optimaalinen sijainti, niin ulomman silmukan tehtävä on etsiä optimaalinen joukko simpleksejä näiden kärkien suhteen. Tässä vaiheessa voidaan verkon elementtejä lisätä ja vähentää tarpeen mukaan. Ulomman silmukan aikana verkkoa muutetaan kolmella operaatiolla, jotka ovat *särmän jako* (edge split), *särmän romahduttaminen* (edge collapse) ja *särmän vaihto* (edge swap). Niiden vaikutukset on esitetty kuvassa 4.3.



Kuva 4.3 Särmän romahdutus, särmän jako ja särmän vaihto.

Jonkin näistä operaatioista suorittamista optimoinnin aikana kutsutaan *siirroksi*. Siirto voi olla joko laillinen tai laiton. Laillinen siirto on sellainen, jonka suorittamisen jälkeen verkon topologia säilyy ennallaan. Särmän jako on aina laillinen, koska se ei voi muuttaa verkon topologiaa. Kaksi muuta siirtoa taas voivat muuttaa topologiaa, joten niitä ei voi suorittaa testaamatta ensin siirron vaikutusta. Ongelmat särmän romahduttamisen kanssa liittyvät tilanteeseen, jossa verkossa on reunoja. Särmä on osa reunaa, jos se kuuluu vain yhteen verkon kolmioon ja kärki on osa reunaa, jos se kuuluu johonkin reunasärmään. Särmä $\{i, j\} \in K$ voidaan poistaa verkosta romahduttamalla vain, jos seuraavat ehdot täyttyvät:

- Jokaiselle sellaiselle verkon kärjelle *k*, jolle on olemassa särmät $\{i,k\}\in K$ ja $\{j,k\}\in K$ on olemassa myös kolmio $\{i,j,k\}\in K$.

- Jos kärjet *i* ja *j* ovat kumpikin reunaan kuuluvia kärkiä, niin silloin myös särmän $\{i,j\}\in K$ on kuuluttava reunaan.

- Verkossa täytyy olla enemmän kuin neljä kärkeä, jos *i* ja *j* eivät kumpikaan kuulu reunaan tai verkossa on yli kolme kärkeä jos joko *i* tai *j* kuuluu reunaan.

Särmän vaihto on laillinen silloin, kun siirron seurauksena syntyvä uusi särmä ei ennestään kuulu verkkoon.

Tavoite on löytää sellainen sarja laillisia siirtoja, joilla alkuperäisen verkon energia saadaan minimoitua. Hoppen ensimmäinen, naivi lähestymistapa oli valita satunnaisesti jokin laillinen siirto ja kokeilla miten sen suorittaminen vaikuttaa energiafunktion arvoon. Jos arvo pienenee, niin siirto jää voimaan ja valitaan seuraava siirto. Jos energia ei jollakin siirrolla pienene niin valitaan uusi siirto. Jos tietyn määrän yrityksiä jälkeen ei ole löydetty energiaa pienentävää siirtoa, niin etsiminen lopetetaan ja tilannetta pidetään optimaalisena. Hoppe kehitti siirtojen valintaan myös heuristisemman lähestymistavan. Siinä pidetään yllä listaa särmistä, jotka todennäköisimmin johtavat energian pienenemiseen, kun niihin sovelletaan jotakin siirtoa. Aluksi listassa on kaikki verkon särmät ja kun jokin särmä valitaan siirtoyrityksen kohteeksi, se poistetaan listasta. Jos siirto hyväksytään, niin särmän ympäristöön kuuluvat särmät lisätään listaan. Tällainen prioriteettilista vähentää sopivan siirron etsimiseen tarvittavaa työtä erityisesti optimoinnin loppuvaiheessa, jolloin suurin osa särmistä ei enää ole sopivia kandinaatteja. Algoritmi 6 antaa pseudokoodiesityksen Hoppen optimointialgoritmista.

Algoritmi 6

Hoppen verkon optimointialgoritmi

Syöte: verkko (V_0, K_0) , alkuperäinen pistejoukko

Tulos: optimoitu verkko

Optimoi kärkien V_0 sijainti verkolle (V_0, K_0) käyttämällä apuna alkuperäistä pistejoukkoa.

do

Muodosta verkko (V', K') suorittamalla verkolle (V_0, K_0) jokin laillinen siirto.

Optimoi kärkien V' sijainti verkolle (V', K') käyttämällä apuna alkuperäistä pistejoukkoa.

if (verkon (V', K') energia on pienempi kuin verkon (V_0, K_0)) {

$$(V_0, K_0) = (V', K')$$

until energia ei enää pienene.

5. Verkkojen muodostaminen ja optimointi käytännössä

Tässä luvussa käsittelen edellä kuvattujen algoritmien toimintaa käytännössä. Algoritmeja testattiin sekä lääketieteellisellä datalla, että geneerisesti tuotetulla testidatalla. Lääketieteellisellä datalla suoritetut testit olivat osa Tampereen teknillisen korkeakoulun ja Tampereen yliopiston yhteisen *M²oBSI* -ryhmän tutkimusprojektia. Verkkojen muodostamista lääketieteellisestä datasta esimerkiksi implisiittisiä pintoja käsittelevillä algoritmeilla on tutkittu runsaasti aikaisemminkin. Tässä työssä tarkoitus oli tutkia miten hyvin yleisillä verkonmuodostusalgoritmeilla kyetään tuottamaan malleja magneettiresonanssikuvien pohjalta muodostetusta syötteestä. Geneerisellä datalla suoritetuilla testeillä tutkittiin lopputuloksien laatua ja yritettiin tuoda esiin algoritmien lähestymistavoista johtuvia eroja lopullisissa malleissa. Algoritmien tehokkuus ei ollut varsinainen tutkimuskohde, vaikka alla käsitelläänkin myös ajoaikoihin liittyviä kysymyksiä.

Testit ajettiin pääsääntöisesti Sunin työasemalla ja palvelimella. Työasema oli malliltaan Sun Ultra Creator 2 3D, jossa on 200 Mhz UltraSparc prosessori ja 256 megatavua muistia. Palvelinkoneessa oli kaksi 300 Mhz prosessoria ja 512 megatavua muistia. Näistä laitteistoista käytetään alla nimityksiä Sun työasema ja Sun palvelin.

5.1 Testeissä käytetyt algoritmit

Verkonmuodostusalgoritmeina käytettiin Hoppen algoritmia ja power crust -algoritmia. Niiden lähestymistapa eroaa tutkituista algoritmeista eniten toisistaan. Crust-algoritmi on lähestymistavaltaan hyvin lähellä power crust -algoritmia ja sitä voidaan pitää yhtenä power crust -algoritmiin johtaneena kehitysvaiheena. Tästä syystä crust-algoritmia ei tutkittu käytännönosuudessa.

Power crust -algoritmista käytettiin algoritmin kehittäjien julkaisemaa vapaasti levitettävää toteutusta. Se käyttää Voronoi-diagrammin ja Delaunay-kolmioinnin muodostamiseen muokattua versiota Ken Clarksonin vapaasti levitettävästä *hull* -ohjelmasta. Hull on helppokäyttöinen, koska se ei vaadi käyttäjältä mitään parametrejä, mutta sen käyttämät algoritmit eivät ole erityisen tehokkaita. Koska power crust -algoritmin antama tulos sisältää myös muitakin monikulmioita kuin kolmioita, niin lopputulos täytyi

vielä muuttaa kolmioverkoksi. Käytetty kolmiointimenetelmä oli naivi, eikä yrittänyt optimoida kolmioiden muotoa. Tässä suhteessa joitakin alla esitettyjä tuloksia voitaneen power crust -algoritmin osalta hieman parantaa. Vaikka algoritmissa onkin muutamia käyttäjän antamia parametreja, niin käytännössä parametrien annettiin olla niiden oletusarvoissa. Myös toteutuksen tekijät raportoivat annettujen oletusarvojen toimivan useimmiten hyvin ja vaativan muutoksia vain tietyissä erikoistapauksissa [Ame01]. Tässä käytetty power crust -toteutus on epädeterministinen. Kahdella peräkkäisellä ajolla saataa samalla syötteellä syntyä hieman toisistaan poikkeavat mallit.

Hoppen algoritmista käytettiin samoin algoritmin tekijän omaa toteutusta. Se on kirjoitettu C++ -kielellä ja sen kääntäminen Sun työasemalla ei ollut aivan ongelmatonta. Eräs Hoppen algoritmin heikkous power crust -algoritmiin verrattuna on, että sen antama lopputulos riippuu huomattavasti enemmän käyttäjän antamista parametreistä. Näiden parametrien säätäminen vaatii tietoa sekä algoritmin toiminnasta, että syötejoukon ominaisuuksista. Erityisesti Hoppen algoritmin vaatima näytetiheys- ja kohinaparametri ($\rho + \delta$) on ongelmallinen. Se ilmaisee pienimmän etäisyyden kahden pinnan "kalvon" välillä. Tämä parametri vaikuttaa siihen, mitkä syötteen osat tulkitaan rei'iksi. Esimerkiksi satunnaisen pallodatan kanssa kävi helposti niin, että liian pienellä tiheysparametrilla valmiiseen malliin syntui reikiä ja liian suurella tiheysparametrilla lopullinen malli oli hyvin karkea. Jos syötteen kuvaamasta kappaleesta tai näytteen tuottamisprosessista ei ole tarkkaa tietoa, niin käytännössä näytetiheysparametrin parhaan arvon etsiminen vaatii sen haarukoimista sopivia arvoja kokeilemalla. Jos syötejoukko on iso ja ajoajat pitkiä, niin tämä vaatii aikaa. Tulosten yhteydessä on Hoppen algoritmin tapauksessa ilmoitettu myös käytetty näytetiheysparametri.

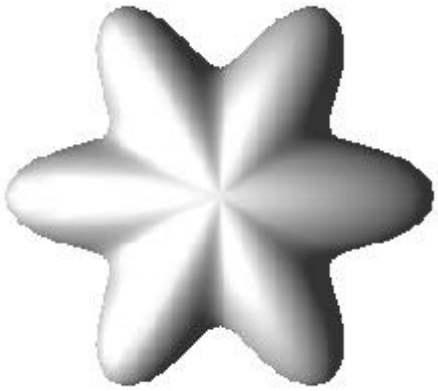
Optimointialgoritmeista kokeiltiin käytännössä Hoppen energiafunktioalgoritmia. Lääketieteellisellä datalla tämä ei valitettavasti ollut mahdollista, sillä käytetty toteutus kieltäytyi käsittelemästä lääketieteellisestä datasta tuotettuja malleja ja pysähtyi "assertion" -virheilmoitukseen. Optimointi onnistui vasta, kun Hoppen algoritmilla tuotettiin malli käyttämällä hyvin suurta näytetiheysparametriä. Tällöin kaikki yksityiskohdat mallista olivat kuitenkin kadonneet ja lopputulos muistutti lähinnä palloa.

Ohjeet käytettyjen toteutusten kääntämisestä ja käyttämisestä löytyvät liitteestä B. Samassa yhteydessä on mainittu verkko-osoitteet, joista toteutukset ovat haettavissa. Lopputuloksena saadut verkot visualisoitiin työtä varten Geomview -nimisellä ilmaisella kolmiulotteisen geometrian visualisointityökalulla. Siihen voi tutustua tarkemmin osoitteessa http://www.geomview.org.

5.2 Käytetty tutkimusaineisto

Aineistona käytettiin sekä aitoa lääketieteellistä dataa, että geneerisesti tuotettua testidataa. Seuraavassa on lyhyesti kuvattu miten testimateriaali tuotettiin ja mitä sen avulla tutkittiin. Algoritmeja testattiin kolmenlaisella datalla: satunnaisella pallodatalla, niin sanotuilla metapalloilla tuotetulla datalla ja jo edellä mainitulla magneettiresonanssidatalla. Pallodata toimi yksinkertaisena perustestinä, jonka avulla kokeiltiin miten algoritmit selviävät helposta, topologialtaan pallomaisesta kohteesta. Samalla voitiin vertailla algoritmien tuottamia verkkoja suhteellisen yksinkertaisella kappaleella, jossa esimerkiksi kolmioiden muoto on kuvista vielä helposti erotettavissa. Metapallodatan tarkoitus oli tuottaa hieman vaikeampi kappale, joka kuitenkin on edelleen topologialtaan pallomainen, mutta jossa on jo mukana sellaisia yksityiskohtia joiden toistumista voidaan tutkia. Lääketieteellinen data toimi todellisena esimerkkinä generointialgoritmien käytöstä ja sen avulla tutkittiin algoritmien soveltuvuutta magneettiresonanssikuvien käsittelyyn.

Satunnainen pallodata tuotettiin yksinkertaisesti arpomalla pallopinnalta haluttu määrä pisteitä. Metapallot taas ovat eräs suljettujen pintojen joukko, joiden avulla voidaan helposti tuottaa kolmiulotteista testidataa [Cxu99]. Erityisesti niillä saadaan aikaan topologialtaan pallomaisia muotoja, joissa on yksityiskohtia. Tarkempi määritelmä metapalloille ja arvot tässä työssä käytetyn syötteen tuottamiseen on annettu liitteessä A. Kuvassa 5.1 on esitetty testeissä käytetty metapalloilla tuotettu pinta.



Kuva 5.1 Testeissä käytetty metapalloilla tuotettu pinta.

Lääketieteellinen data muodostettiin magneettiresonanssikuvien pohjalta Tampereen teknillisen korkeakoulun signaalinkäsittelylaitoksella. Magneettiresonanssikuvaus (MR -kuvaus) on erityisesti lääketieteessä käytetty menetelmä, jonka avulla saadaan tuotet-

tua kuvia ihmiskehon sisältä. Menetelmän avulla tuotettu kuvadata saadaan vokselimuodossa siten, että vokseleihin on liitetty kudostyyppiä kuvaava intensiteettiarvo. Tässä työssä käytetty data muodostettiin aivoista otetun kolmiulotteisen MR -datan perusteella. Kuvadata segmentoitiin, eli jaoteltiin alueisiin eri kudostyyppien mukaan. Tässä tapauksessa ne olivat aivojen valkoinen ja harmaa aine, sekä selkäydinneste. Kaikki muut kudostyypit, kuten kallon luut jätettiin pois. Tämä segmentointiprosessi on täysin automaattinen, mutta ei kuitenkaan aivan vedenpitävä, sillä kohinasta, kuvien resoluutiosta ja kuvattavan kohteen muodosta johtuen jotkut vokselit saattavat luokittua väärin. Tällaiset tilanteet täytyi siivota pois käsin. Tämän jälkeen datasta etsittiin ne pisteet, jotka kuuluvat haluttuun pintaan, joka tässä tapauksessa oli aivojen harmaan ja valkoisen aineen välinen rajakerros. Näin saatua pistejoukkoa käytettiin sitten syötteenä tutkituille algoritmeille.

Tavoitteena oli ensinnäkin selvittää minkä tasoisia kolmiulotteisia malleja tällä hetkellä tunnetuilla yleispätevillä algoritmeilla voidaan tuottaa MR -kuvien perusteella muodostetusta datasta. Toisena tavoitteena oli selvittää voidaanko hankalasta MR -kuvadatasta tuottaa topologialtaan pallomaisia malleja tilanteessa, jossa tiedetään että alkuperäinen pinta on topologialtaan pallomainen. Verkonmuodostuksen kannalta harmaan ja valkoisen aineen välinen rajakerros on muodoltaan hyvin vaikea, koska se sisältää poimuja ja syviä uurteita. Tätä tilannetta vaikeuttaa vielä MR -kuvien suhteellisen alhainen resoluutio ja mahdolliset kohinasta sekä segmentointiprosessista johtuvat virheet. Mallien laatua arvioitiin puhtaasti verkonmuodostuksen kannalta eikä tarkoitus ollut tutkia niiden mahdollista lääketieteellistä käyttökelpoisuutta. Lisäksi lääketieteellisellä datalla tutkittiin, miten algoritmit kykenevät käytännössä käsittelemään hyvin suuria syötteitä.

5.3 Tulosten arvioinnissa käytetyt metriikat

Tuotettuja malleja arvioitiin sekä silmämääräisesti että muutamien metriikoiden avulla. Ensinnäkin malleista ilmoitetaan kärkien, särmien ja kolmioiden lukumäärä sekä Eulerin karakteristika. Kuten edellä esittettiin, Eulerin karakteristikalla saadaan tietoa kappaleen topologiasta ja voidaan esimerkiksi tutkia syntyykö pallomaisen topologian omaavasta kappaleesta topologialtaan pallomainen verkko. Jos pallomaisen topologian omaavan kappaleen perusteella rakennetun verkon Eulerin karakteristika on pienempi kuin 2, niin verkon voidaan ajatella olevan sitä huonompi, mitä pienempi Eulerin karakteristika on. Mitä enemmän arvo eroaa luvusta 2, niin sitä enemmän verkonmuodostusalgoritmi on tuottanut tilanteita, joissa kärkien, särmien ja kolmioiden suhde ei

vastaa pallomaista kohdetta. Lisäksi mallien kolmioille laskettiin kolmion pisimmän ja lyhimmän sivun suhde ja laskettiin näiden suhteiden keskiarvo. Se kertoo miten lähellä tasasivuista kolmiota mallin kolmiot keskimäärin ovat.

5.4 Tulokset

Kaikkien testiajojen tulokset on koottu taulukkoon 1. Taulukossa on esitetty sekä syötejoukkojen koot että kohdassa 5.3 esitetyt metriikat eri algoritmeilla tuotetuista malleista.

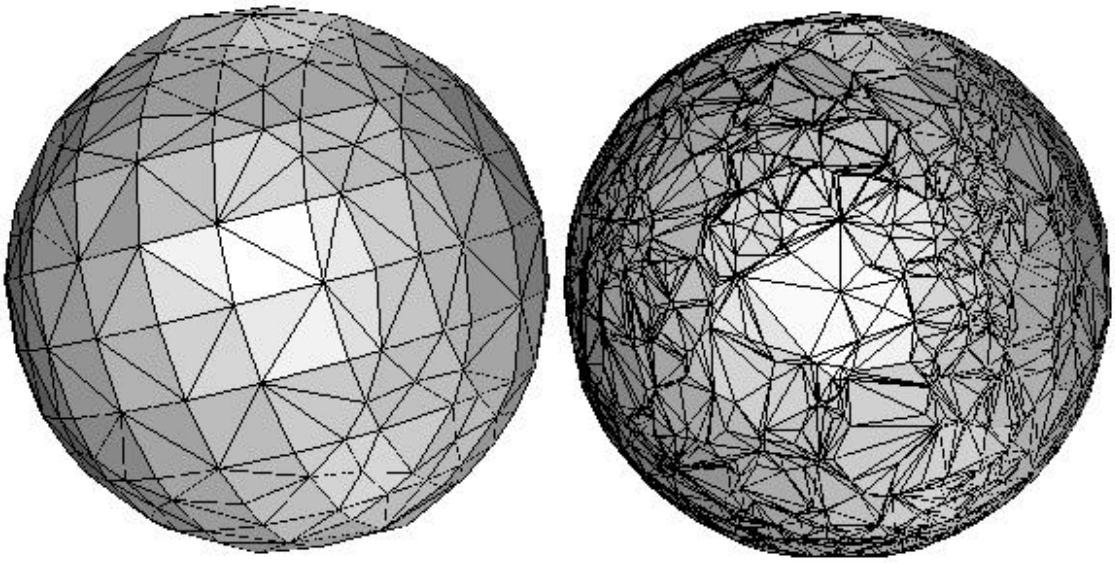
Taulukko 1. Taulukossa käytetään seuraavia merkintöjä: **UC10** ja **UC11** ovat MR –kuvien perusteella tuotettuja joukkoja. Tunnus **SP1000** on satunnaisesti tuotettua pallodataa ja tunnus **STAR50** on metapallomenetelmällä tuotettu joukko. **Syöte** ilmaisee syötejoukon koon. **Kärjet**, **särmät** ja **kolmiot** kertovat vastaavien elementtien lukumäärän verkossa. Tunnus **E**k on Eulerin karakteristika ja tunnus **Er** on kolmioiden sivujen suhteiden keskiarvo. Tunnus r + d kertoo Hoppen algoritmin tapauksessa käytetyn näytetiheysparametrin.

	Syöte	**Kärjet**	**Särmät**	**Kolmiot**	**E**k	**Er**	r + d
Hoppen algoritmi							
UC10	89691	73456	216338	140297	-2585	0.4165	0.011
UC11	93868	47481	139153	89848	-1824	0.4172	0.015
SP1000	1000	388	1158	772	2	0.3482	0.115
STAR50	2500	1604	4678	2996	-78	0.3984	0.062
Power crust							
UC10	89691	403360	1214081	809280	-1441	0.1608	
UC11	93868	430530	1295283	863329	-1424	0.1621	
SP1000	1000	4910	14724	9816	2	0.1517	

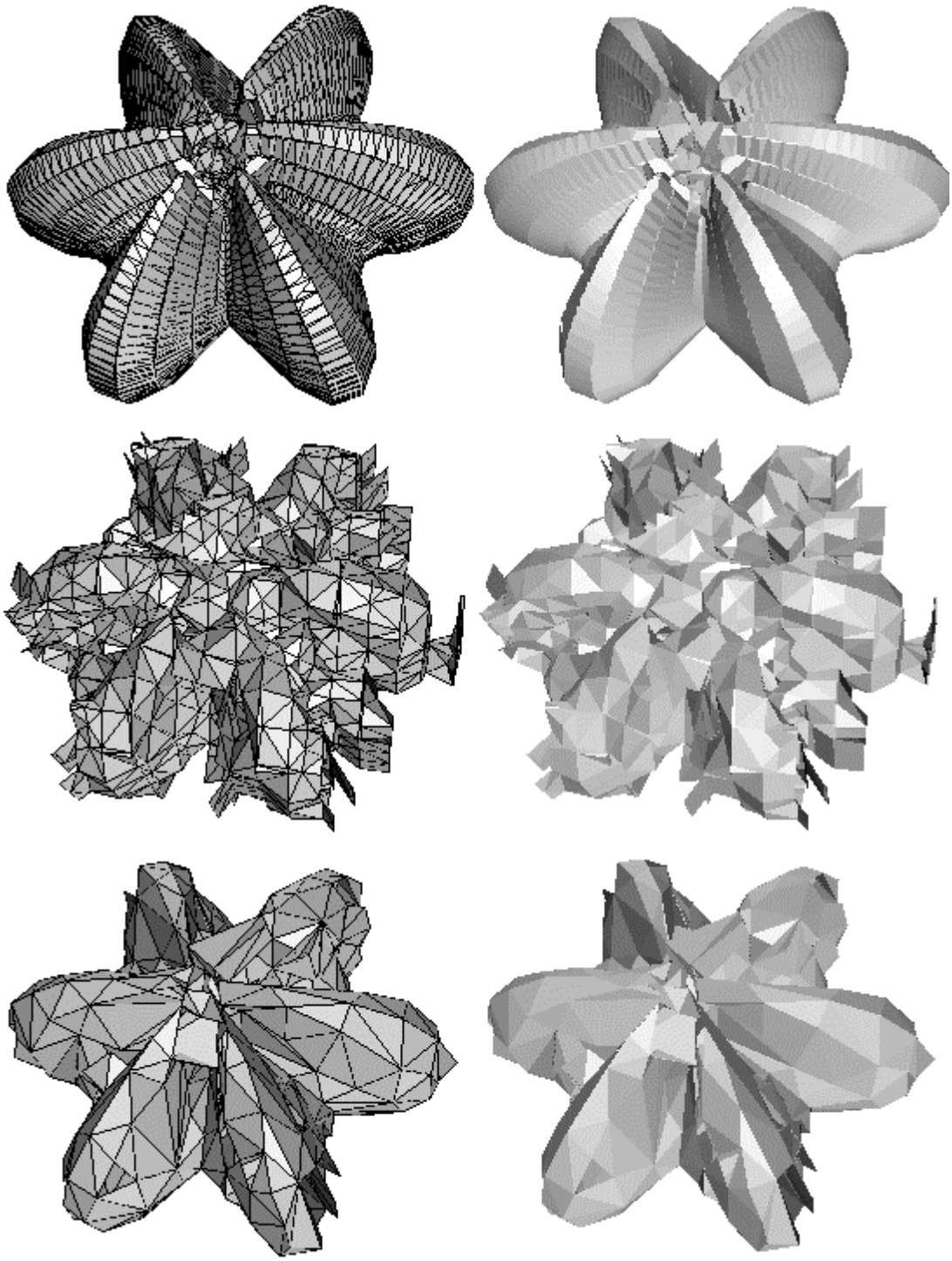
STAR50	2500	20196	60738	40492	-50	0.0750	
Hoppe, optimoitu							
SP1000	1000	200	594	396	2	0.5955	0.115
STAR50	2500	534	1662	1050	-78	0.3668	0.062

Tuloksia on lisäksi havainnollistettu myös kuvin. Jokaisesta syötetyypistä on esitetty luva, jonka avulla on mahdollista vertailla eri algoritmien tuottamia malleja. Yhdessä kuvassa on eri menetelmillä tuotettuja malleja samasta kulmasta ja samalta etäisyydeltä kuvattuna. Kolmioiden reunat on piirretty näkyviin, jotta erot niiden muodossa, ryhmityksessä ja lukumäärässä on helppo havaita. Osassa kuvista on lisäksi esitetty myös varjostettu malli, jossa kolmioiden reunoja ei ole piirretty näkyviin. Magneettiresonanssisyötteestä tuotettuja malleja esittävissä kuvissa kolmioiden reunoja ei ole piirretty näkyviin mallien monimutkaisuudesta johtuen.

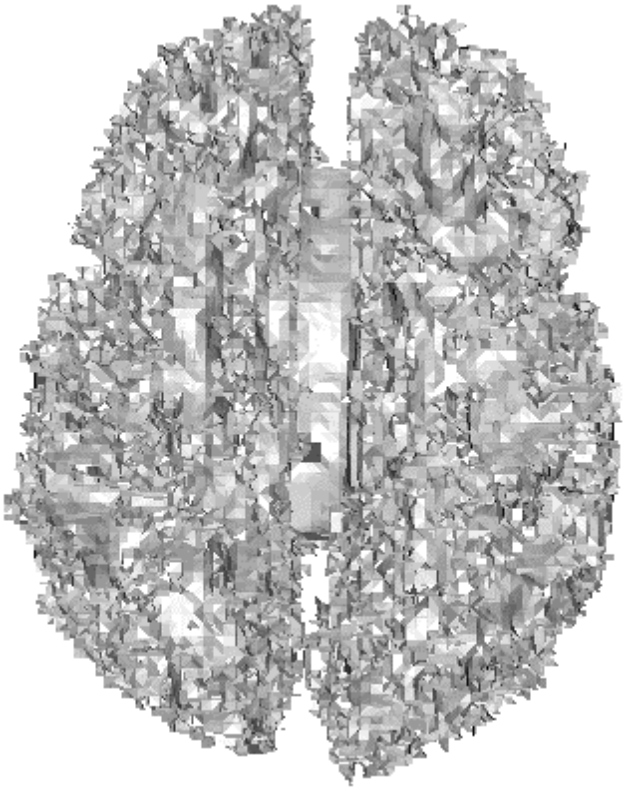
Kuvassa 5.2 on esitetty vierekkäin satunnaisesta pallodatasta muodostetut mallit kummallakin verkonmuodostusalgoritmilla tuotettuna. Kuva esittää yksinkertaisen perustilanteen hyvin helpolla syötteellä. Kuva 5.3 esittää metapallodatasta tuotetut mallit. Kuvassa on verkonmuodostusalgoritmien perusversioilla tuotettu malli, sekä lisäksi Hoppen optimointialgoritmilla optimoitu malli. Kuvissa 5.4 ja 5.5 on magneettiresonannsidatasta muodostetut mallit. Kuvassa 5.6 on edellisten kuvien malleista suurennettu yksityiskohta. Kuva 5.7 havainnollistaa optimoimattoman ja optimoidun mallin välisiä eroja.



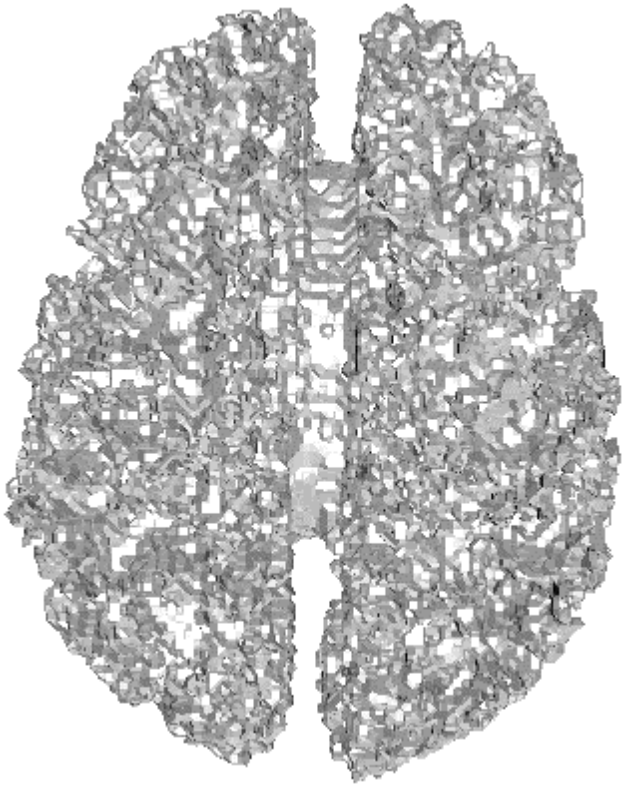
Kuva 5.2 Satunnaisesta pallodatasta tuotetut verkot. Vasemmalla Hoppen algoritmilla tuotettu verkko ja oikealla vastaava power crust -algoritmilla tuotettu verkko.



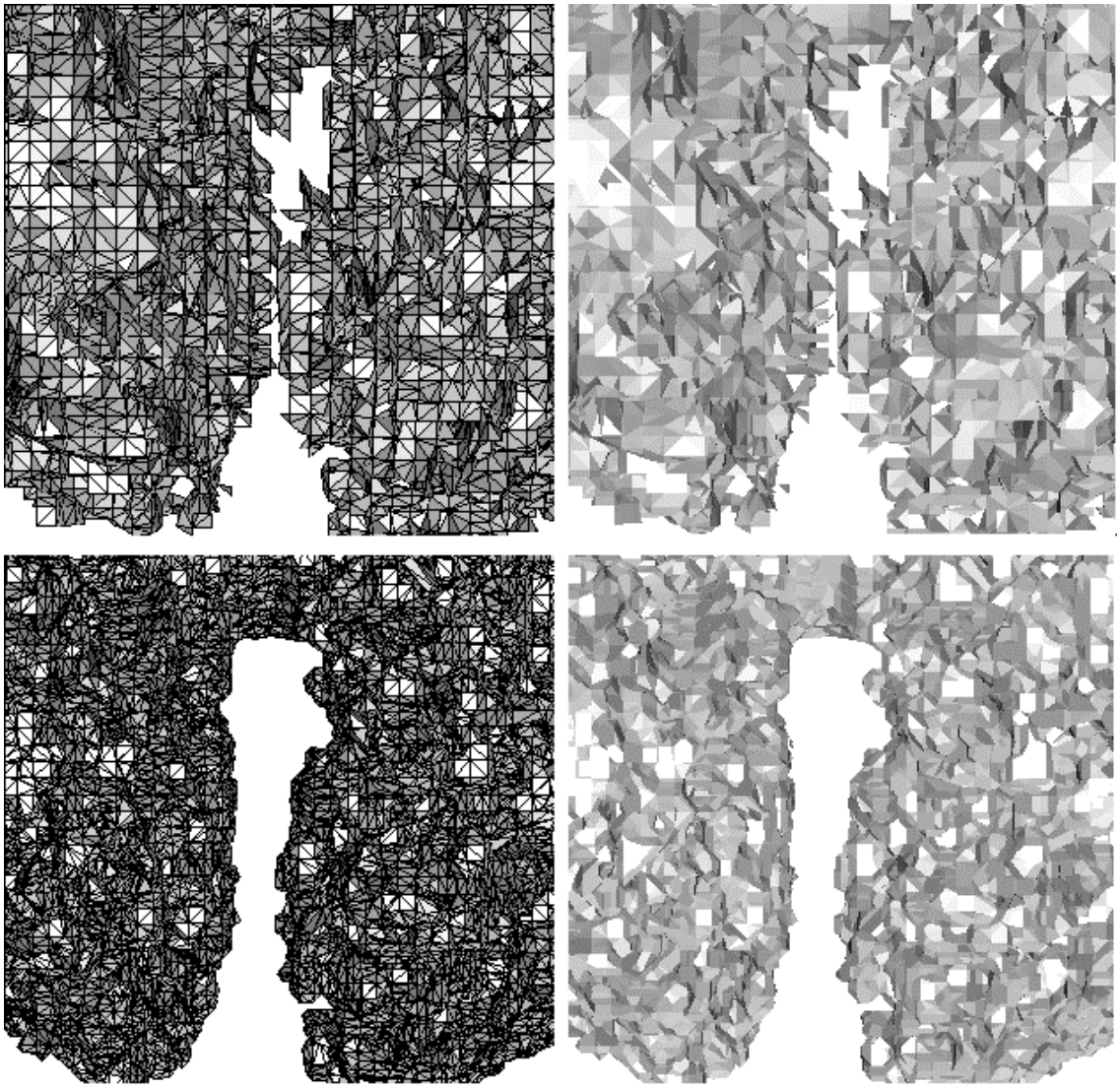
Kuva 5.3 Metapallodatasta tuotetut mallit. Ylimpänä power crust -algoritmillla ja keskellä Hoppen algoritmillä tuotettut mallit. Alimpana optimoitu versio Hoppen algoritmillä tuotetusta mallista.



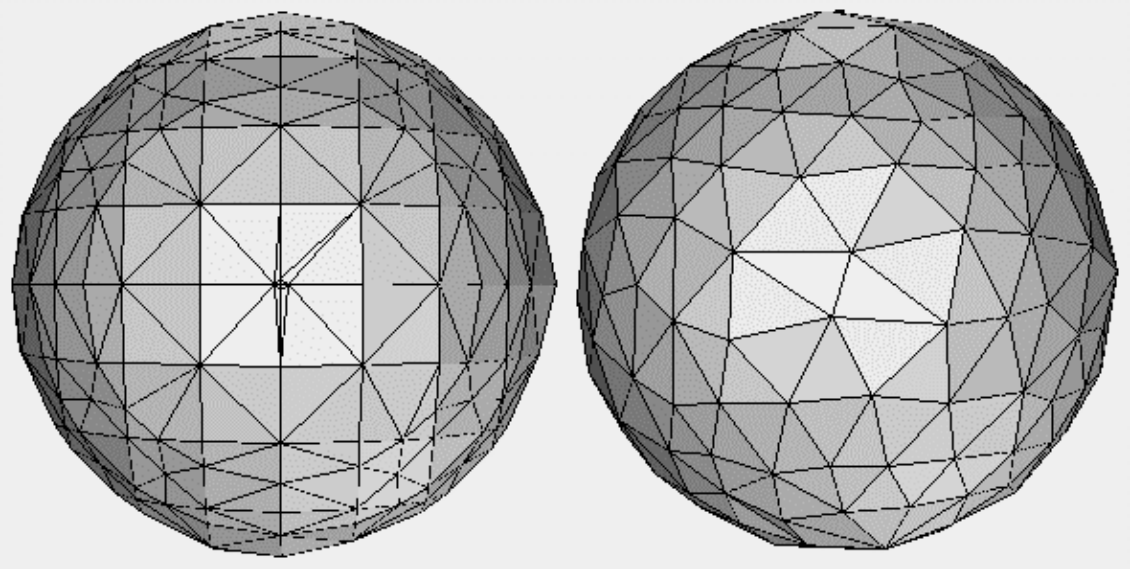
Kuva 5.4 Hoppen algoritmilla tuotettu malli magneettiresonanssidatasta.



Kuva 5.5 Power crust -algoritmilla tuotettu malli magneettiresonanssidatasta.



Kuva 5.6 Suurennokset magneettiresonanssidatasta tuotetuista malleista. Ylemmällä rivillä Hoppen algoritmilla tuotettu malli ja alemmalla rivillä power crust -algoritmillä tuotettu malli.



Kuva 5.7 Vasemmalla Hoppen algoritmilla tuotettu malli satunnaisesta pallodatasta (SP1000) ja oikealla siitä energiafunktiomenetelmällä optimoitu versio.

5.5 Tulosten pohdinta

Taulukon 1 perusteella merkittävin ero verkonmuodostusalgoritmien välillä on lopputuloksen monimutkaisuudessa. Power crust -algoritmi tuottaa samoista syötteistä huomattavasti Hoppen algoritmia monimutkaisempia malleja. Kolmioiden määrä voi power crust -algoritmin tuottamassa mallissa olla yli kymmenkertainen Hoppen algoritmin tuottamaan malliin verrattuna. Kärkien määrä power crust -algoritmilla tuotetuissa malleissa näyttää olevan 4-5 kertaa suurempi kuin alkuperäisen syötejoukon pisteiden lukumäärä. Kaikki syötejoukkojen kuvaamat pinnat olivat topologialtaan pallomaisia, mutta vain satunnaisesta pallodatasta algoritmit kykenivät tuottamaan topologialtaan pallomaisia verkkoja. Hoppen algoritmin tuottamat kolmiot olivat muodoltaan huomattavasti lähempänä tasasivuista kolmiota, kuin power crust -algoritmilla tuotetut. Hoppen optimiontialgoritmin käyttö paransi entisestään kolmioiden muotoa.

Suuri ero monimutkaisuudessa näkyy selvästi arvioitaessa verkkojen laatua visuaalisesti. Kuvan 5.2 satunnaisesta pallodatasta tuotetuisssa malleissa havainnollistuu hyvin erot algoritmien lähestymistavassa. Hoppen algoritmin tuottaman verkon kolmiot ovat hyvin säännöllisen muotoisia. Datan jakaminen kuutioihin ennen sen antamista lopullisen verkon tuottavalle marching cubes –algoritmille on selvästi nähtävissä. Power crust -algoritmi tuottaa huomattavasti enemmän kolmioita ja lisäksi kolmioiden muoto on hyvin vaihteleva. Suuri osa kolmioista on muodoltaan pitkiä ja kapeita. Tähän tosin

vaikuttaa myös käytetty kolmiointimenetelmä siinä vaiheessa, kun algoritmin tuottama malli muutetaan kolmioverkoksi. Kuvan esittämässä tilanteessa power crust -algoritmi on tuottanut liiankin monimutkaisen mallin, pallon muoto ei olennaisesti parannu huomattavasti suuremmasta kolmioiden määrästä huolimatta. Muilla tutkituilla syötteillä tilanne on toinen.

Kuvassa 5.3 käytetty metapallomenetelmällä tuotettu syöte osoittautui yllättävän hankalaksi. Kuvassa ylimmällä rivillä on power crust -algoritmilla tuotettu malli, keskimmäisellä rivillä Hoppen algoritmilla tuotettu malli ja alimmalla rivillä edellissestä optimoitu versio. Optimointiin käytettiin Hoppen energiafunktioihin perustuvaa menetelmää. Tässä tapauksessa power crust -algoritmi onnistui tuottamaan selvästi parhaan mallin, mutta silläkin oli suuria vaikeuksia kappaleen keskustassa. Tähden keskusta, jossa sakarat yhtyvät on hajonnut pahasti. Tässä kohdassa Voronoi-solujen muoto ei enää noudata algoritmin asettamia vaatimuksia ja se ei enää osaa päätellä kuuluvatko painotetun Voronoi-diagrammin solut sisä- vai ulkopuolelle. Power crust -algoritmin kanssa kokeiltiin myös luvussa 3 mainittua terävät kulmat säilyttävää tekniikkaa, mutta silti tilanne ei oleellisesti parantunut, vaan tähden keskusta oli edelleen epämuodostunut. Epämuodotunutta aluetta voitaisiin pienentää kasvattamalla näytetiheyttä tähden keskustassa.

Hoppen algoritmi selvisi metapallodatasta vielä huonommin. Sopivan näytetiheysparametrin etsiminen tälle syötteelle osoittautui hankalaksi, ja lopulta parhaaksi jääneeseen malliinkin jäi reikiä. Pelkällä Hoppen verkonmuodostusalgoritmilla muodostettu malli (kuvan toinen rivi) muistuttaa korkeintaan epämääräisesti alkuperäistä kappaletta. Tämä syöte tuo hyvin esille sen, että Hoppen algoritmi on suunniteltu toimimaan yhteistyössä Hoppen optimointialgoritmin kanssa. Optimoitu versio muistuttaa jo huomattavasti enemmän alkuperäistä kappaletta. Hoppen verkonmuodostusalgoritmi onkin tarkoitettu tuottamaan lähtökohta, jonka perusteella optimointialgoritmi luo lopullisen mallin yhdessä alkuperäisen syötejoukon kanssa. Otimoitu versiokaan ei kuitenkaan pääse lähellekkään power crust -algoritmin tuottamaa mallia, vaan on siihen verrattuna karkea ja epämuodostunut. Kummatkaan algoritmit eivät onnistuneet säilyttämään alkuperäisen pinnan pallomaista topologiaa. Power crust -algoritmin tapauksessa pallomainen topologia saatiin kyllä säilymään säätämällä sopivasti näytetiheysparametria, mutta tällöin verkko ei enää juuri muistuttanut alkuperäistä kappaletta.

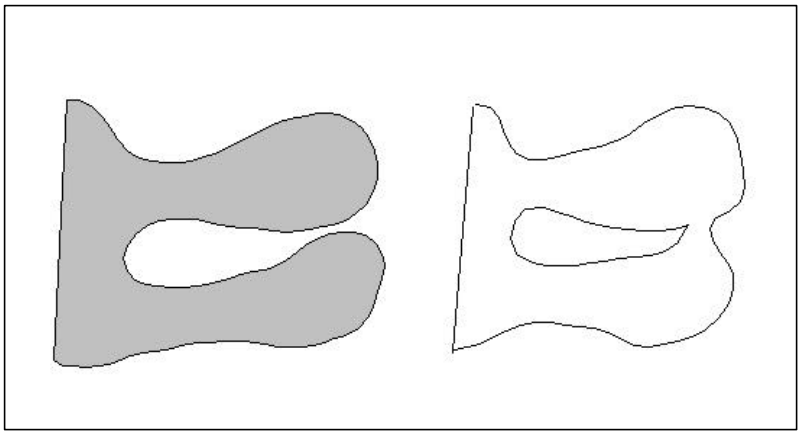
Optimoiduissa malleissa verkon topologia säilyy, vaikka kolmioiden lukumäärä on pudonnut toisessa tapauksessa noin puoleen ja toisessa tapauksessa noin kolmasosaan al-

kuperäisestä. Kolmioiden huomattavasta vähentämisestä huolimatta optimoidut mallit kuvaavat kuitenkin alkuperäistä pintaa paremmin kuin alkuperäiset mallit (esimerkiksi kuva 5.3). Kolmioiden keskimääräinen muoto on olennaisesti muuttunut optimoinnin seurauksena vain satunnaisen pallodatan tapauksessa.

Kummatkin algoritmit selvisivät hankalasta magneettiresonanssidatasta periaatteessa hyvin. Kuvassa 5.4 on esitetty power crust -algoritmilla tuotettu malli ja kuvassa 5.5 vastaava Hoppen algoritmilla tuotettu malli. Mallit on muodostettu algoritmien perusversioilla ilman, että niitä on käsitelty tai optimoitu jälkeenpäin. Saadut mallit muistuttavat silmämääräisesti alkuperäistä kohdetta ja mitään suuren mittakaavan virheitä ei syntynyt. Esimerkiksi aivopuoliskot ja niitä yhdistävä aivokurkiainen ovat selvästi toistuneet erillisinä kokonaisuuksina. Power crust -algoritmi tuotti odotetusti huomattavasti monimutkaisempia malleja kuin Hoppen algoritmi. Monimutkaisimmat sillä tuotetut mallit ylittävät miljoonan kolmion rajan. Power crust -algoritmin tuottamassa mallissa aivopuoliskojen välinen tila säilyi avoimena ja reunat ovat hyvin tasaiset. Hoppen algoritmin tuottamassa mallissa sen sijaan oikean ja vasemman aivopuoliskon reunat rönsyilevät ikävästi ja yhdessä kohdassa jopa koskettavat toisiaan. Tämä ei ole seurausta pienemmästä kolmioiden määrästä, vaan Hoppen algoritmi on selvästi tässä kohtaa epäonnistunut pinnan etsinnässä. Ilmeisesti Hoppen algoritmiin liittyvä *k*-parametri on ollut liian suuri, jolloin väärän aivopuoliskon näytteitä on tullut mukaan tangenttitason muodostukseen. Vaikka Hoppen optimointialgoritmilla mallin laatua voitaisiin parantaa, se on kuitenkin suunniteltu säilyttämään verkon topologia, joten tällaista tilannetta se ei pysty korjaamaan. Kuvan 5.6 esittämässä suurennoksessa tilanne näkyy selvemmin. Kuvasta on myös nähtävissä power crust -algoritmin huomattavasti parempi kyky toistaa pienemmät yksityiskohdat. Siinä missä aivopuoliskojen välisen tilan reuna on Hoppen algoritmin tuottamassa mallissa osittain umpeutunut ja rönsyilevä, se on power crust -algoritmin mallissa hyvin selkeä ja tasainen.

Kummatkaan algoritmit eivät kyenneet tuottamaan magneettiresonanssidatasta topologialtaan pallomaisia malleja, vaikka syötejoukko kuvasikin topologialtaan pallomaista kappaletta. Tämä oli odotettavissa, eikä johdu suoranaisesti käytettyistä algoritmeista. Syy epäonnistumiseen on sekä magneettiresonanssikuvien vielä suhteellisen heikossa resoluutiossa, että tavassa, jolla kuvadata prosessoidaan ennen kuin siitä saadaan algoritmeille sopiva syöte. Kuvadataan syntyy esimerkiksi aivokuvien kohdalla kohtia, joissa kaksi aivokudoksen poimua on hyvin lähellä toisiaan tai jopa koskettaa toisiaan, jolloin niiden juurelle syntyy umpinainen alue. Tätä on havainnollistettu kuvassa 5.8. Kuvassa vasemmalla on tilanne, jossa poimun päät ovat hyvin lähellä toisiaan ja oikealla tästä virheellisesti etsitty reuna. Tämän ilmiön

seurauksena myös lopullisessa mallissa poimujen päät ovat sulautuneet yhteen. Tällaisissa kohdissa vaadittaisiin hyvin korkea näytetiheys, ennenkuin nykyiset algoritmit kykenisivät rakentamaan pinnan oikein.



Kuva 5.8 Poimu -ongelma.

Kuvassa 5.7 on esitetty vierekkäin Hoppen algoritmilla tuotettu malli satunnaisesta pallodatasta ja sen pohjalta Hoppen energiafunktiomenetelmällä optimoitu malli. Optimoidussa versiossa on noin puolet vähemmän kolmioita kuin optimoimattomassa mallissa. Optimoidun version kolmiot ovat selvästi muodoltaan homogeenisempia verrattuna alkuperäiseen malliin ja esimerkiksi alkuperäisessä mallissa keskellä näkyvät pitkät ja kapeat kolmiot ovat hävinneet.

Power crust -algoritmin ongelma on suuri muistintarve. Noin 90000 pisteen käsittely Sun työasemalla vei päiviä, kunnes kulutti loppuun kaiken koneelle varatusta 600 megatavun heittovaihtomuistista. Ensimmäinen neljäsosa työstä suoritettiin muutamassa minuutissa, mutta työmuistin loputtua algoritmin suoritus hidastui huomattavasti. Käytännössä vaaditaan ainakin 512 megatavua muistia yli 100000 pisteen näytteiden käsittelyyn. Sun palvelimella 90000 pisteen syötteen käsittely vei 4-5 tuntia. Jos käytössä olisi riittävästi muistia, niin ajoajat olisivat tämänkin kokoisella syötteellä alle puolen tunnin luokkaa. Käytetyn toteutuksen tekijät raportoivat sen kykenevän käsittelemään 30000 pisteen joukon noin kuudessa minuutissa 400 Mhz Sun työasemalla. Ajoaikoja voitaisiin edelleen parantaa käyttämällä pohjalla tehokkaampia algoritmeja Delaunay-kolmioinnin ja Voronoi-diagrammin laskemiseen. Suurin ongelma on kuitenkin se, että käytetty power crust -toteutus antaa lopputuloksen, jossa sama kärki saattaa esiintyä verkossa useita kertoja ja verkko saattaa sisältää särmiä, joiden pituus on nolla. Nämä moninkertaiset esiintymät ja tyhjät särmät täytyy siivota pois. Tämä kaikki sai aikaan sen, että kuvatulla laitteistolla ja tällä toteutuksella valmiin mallin tuottaminen kesti pahimmillaan 1-2 vuorokautta.

Hoppen algoritmi toimii huomattavasti pienemmälllä määrällä muistia. Ajoajat olivat Sun työasemalla muutaman minuutin luokkaa. Siinä missä power crust -algoritmilla voi melko vapaasti luottaa eri parametrien oletusarvoihin, vaatii Hoppen algoritmi huomattavasti enemmän yrityksiä ja parametrien säätöä ennenkuin se saadaan tuottamaan paras mahdollinen malli.

6. Yhteenveto

Tässä työssä tutkin kolmiulotteisten pintojen kuvaamista verkoilla, verkkojen tuottamista automaattisesti ja niiden optimointia. Tavoitteena oli etsiä mahdollisimman yleispätevä menetelmä verkon tuottamiseen järjestämättömästä kolmiulotteisesta näytejoukosta. Tarkemmassa tarkastelussa oli kolme keskeisintä yleistä verkonmuodostusalgoritmia. Kaksi näistä algoritmeista perustui pohjimmiltaan Voronoi-solujen napojen ja kappaleen keskiakselin välisen suhteen hyödyntämiseen ja varsinainen niitä erottava tekijä oli strategia, jolla verkko tämän tiedon perusteella rakennetaan. Toinen menetelmistä käyttää Voronoi-suodatukseksi kutsuttua menetelmää ja toinen muodostaa verkon luokittelemalla muunnellun Voronoi-diagrammin soluja sisä- ja ulkopuolisiin soluihin ja etsimällä niiden välisen rajakerroksen. Kolmas tutkittu lähestymistapa pyrkii rakentamaan syötteen perusteella pintaa kuvaavan etäisyysfunktion, jonka perusteella lopullinen verkko muodostetaan käyttämällä implisiittisiä pintoja käsitteleviä algoritmeja.

Lisäksi tarkastelin verkkojen optimointia. Kahta erilaista lähestymistapaa tutkittiin tarkemmin. Toinen pyrkii vähentämään kolmioiden lukumäärää tutkimalla verkon paikallista geometriaa jonkin kärjen ympäristössä ja poistamalla kärjen verkosta mikäli paikallinen geometria ei kärsi liikaa. Toinen menetelmä perustui verkon laatua kuvaavan energiafunktion minimointiin ja alkuperäisen syötejoukon pisteiden hyödyntämiseen.

Osaa tutkituista algoritmeista kokeiltiin myös käytännössä. Verkkoja tuotettiin sekä geneerisesti tuotetusta datasta että lääketieteellisestä magneettiresonansidatasta. Tutkitut algoritmit olivat power crust –algoritmi ja Hoppen algoritmi. Tulosten perusteella algoritmit tuottivat hyvin erilaisia verkkoja. Suurin ero oli verkkojen monimutkaisuudessa. Menetelmiä ei kuitenkaan voida asettaa paremmuusjärjestykseen, sillä niiden solveltuvuus eri tilanteisiin riippuu lopputuloksen vaatimuksista. Power crust -algoritmi tuottaa huomattavasti monimutkaisempia malleja kuin Hoppen algoritmi. Toisaalta power crust -algoritmin tuottamat pinnat vastaavat Hoppen algoritmia paremmin alkuperäistä kappaletta erityisesti monimutkaisilla syötteillä. Hoppen algoritmin tuottamat kolmiot ovat power crust -algoritmin tuottamiin verrattuna muodoltaan homogeenisempia ja geometrisiltä ominaisuuksiltaan tasapainoisempia. Lisäksi Hoppen algoritmin tuottamia malleja voidaan edelleen parantaa Hoppen optimointialgoritmin avulla. Hoppen algo-

ritmi sellaisenaan on tarkoitettu verkonmuodostusprosessin ensimmäiseksi vaiheeksi, jonka tuloksia seuraavassa vaiheessa muokataan optimointialgoritmilla.

Power crust -algoritmi on käyttäjäystävällisempi, kun taas Hoppen algoritmi vaatii enemmän tietämystä algoritmin toiminnasta ja huomattavasti enemmän parametrien säätämistä parhaan lopputuloksen saavuttamiseksi. Hoppen algoritmi kykenee myös perusmuodossaan käsittelemään pintoja, joissa on reikiä ja reunoja, kun taas power crust -algoritmi tuottaa aina "vesitiiviin" pinnan.

Magneettiresonanssidatalla parhaaseen lopputulokseen päästiin power crust –algoritmilla. Vaaditaan kuitenkin lisää kehitystyötä sekä magneettiresonanssisyötteiden tuottamisen että verkonmuodostusalgoritmien alueella, ennen kuin magneettiresonanssidatasta kyetään muodostamaan topologisesti oikeita verkkoja.

Viiteluettelo

[Alb95] Lyuba Alboul ja Ruud van Damme. *Polyhedral Metrics in Surface reconstruction: Tight Triangulations.* University of Twente, Dep. of Applied Mathematics, Technical Report, 1995.

[Ame98] Nina Amenta, Mashall Bern, Manolis Kamvysselis. *A New Voronoi-Based Surface Reconstruction Algorithm.* SIGGRAPH 98 Conference Proceedings, 1998.

[Amb98] Nina Amenta, Marshall Bern ja David Eppstein. *The Crust and the Beta Skeleton. Combitorial Curve Reconstruction.* Graphical Models and Image Processing, vol **60(2)**, sivut 125-135, 1998.

[Ame99] Nina Amenta ja Mashall Bern. *Surface Reconstruction by Voronoi filtering.* Discrete and Computational Geometry, vol **22**, sivut 481-504, 1999.

[Ame01] Nina Amenta, Sunghee Choi ja Ravi Krishna Kolluri. *The Power Crust.* Tällä hetkellä saatavilla vain Internetistä osoitteesta http://www.cs.utexas.edu/users/amenta/pubs/sm.pdf. Tullaan julkaisemaan vuoden 2001 ACM Symposium of Solid Modelling and Applications -julkaisussa.

[Boi84] Jean-Daniel Boissonnat. *Geometric Structures for three-dimensional shape reconstruction.* ACM Transaction on Graphics, vol **3**, sivut 266-286, 1984.

[Dom98] Julien Dompierre, Paul Labbé, Francois Guibault ja Ricardo Camarero. *Proposal of Benchmarks for 3D Unstructured Tetrahedral Mesh Optimization.* Proceedings of 7th International Meshing Roundtable, 1998.

[Ede94] Herbert Edelsbrunner ja Ernst P. Mücke. *Three-dimensional Alpha Shapes*. ACM Transactions on Graphics, vol **13,** 1994.

[For95] Steven Fortune. *Voronoi diagrams and Delaunay triangulations.* Handbook of Discrete and Computational Geometry, sivut 377-388, CRC Press, New York, 1995.

[Gol00] Christopher Gold ja Jack Snoeyink. *A One-Step Crust and Skeleton Exctraction Algorithm*, 2000.

[Hil97] Adrian Hilton ja John Illingworth. *Marching Triangles: Delaunay Implicit Surface Triangulation.* CVSSP Technical Report 01, tammikuu 1997.

[Hop92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, Werner Stuetzle. *Surface Reconstruction from Unorganized Points*. Computer Graphics (SIGGRAPH 1992 Proceedings), sivut 71-78, 1992.

[Hop94] Hugues Hoppe. *Surface Reconstruction from Unorganized Points.* Väitöskirja. University of Washington, 1994.

[Hop97a] Hugues Hoppe, Tony DeRose, Tom Duchamp, John MacDonald ja Werner Sctuetzle. *Mesh Optimization.* Multiresolution Surface Modeling (Proceedings of SIGGRAPH '97), 1997.

[Hop97b] Hugues Hoppe. *Progressive Meshes*. Multiresolution Surface Modeling (Proceedings of SIGGRAPH '97), 1997.

[Hop97c] Hugues Hoppe. *View-Dependent Refinement of Progressive Meshes.* Multiresolution Surface Modeling (Proceedings of SIGGRAPH '97), 1997.

[Knu01] Patrick M. Knupp. *Algebraic mesh quality metrics.* Proceedings of the 9th International Meshing Roundtable, sivut 173-183, 2001.

[Lor87] William E. Lorensen ja Harvey E. Cline. *Marching Cubes: A High Resolution 3D Surface Construction Algorithm.* Computer Graphics (Proceedings of SIGGRAPH '87), vol **21(4)**, sivut 163-169, 1987.

[Mon94] Claudio Montani, Riccardo Scateni, Roberto Scopigno. *Discretized Marching Cubes.* Visualization '94 Proceedings, sivut 281-287. IEEE Computer Society, IEEE Computer Society Press, 1994.

[Nin93] Paul Ning ja Jules Bloomenthal. *An Evaluation of Implicit Surface Tilers*. IEEE Computer Graphics & Applications, vol **13(6)**, sivut 33-41, 1993.

[Sch97] William J. Schroeder, Jonathan A. Zarge, William E. Lorensen. *Decimation of triangle meshes.* Multiresolution Surface Modeling (Proceedings of SIGGRAPH '97), 1997.

[Wat93] Alan Watt. *3D Computer Graphics*. Addison-Wesley, 1993.

[Cxu99] Cengyang Xu. *Deformable Models with Application to Human Celebral Cortex Reconstruction from Magnetic Resonanse Images.* Väitöskirja. Johns Hopkins University, 1999.

Liitte A

Metapallot

Metapallot ovat joukko suljettuja kolmiulotteisia pintoja, joiden avulla voidaan helposti tuottaa symmetristä, topologialtaan pallomaista testidataa. Seuraavassa metapallojen tarkempi määrittely. Se on peräisin Chengyang Xun väitöskirjasta [Cxu99].

Määritellään $\bar{x} = (x, y, z)$, $a = (a_x, a_y, a_z)$, $b = (b_x, b_y, b_z)$, $m = (m_x, m_y, m_z)$ ja $n = (n_{x,} n_y, n_z)$. Tällöin metapallo $\bar{x}(q, f)$ määritellään seuraavasti

$$x = (a_x + b_x \cos(m_x q)\cos(n_x f))\sin(q)\cos(f)$$
$$y = (a_y + b_y \cos(m_y q)\cos(n_y f))\sin(q)\sin(f)$$
$$z = (a_z + b_z \cos(m_z q)\cos(n_z f))\cos(q)$$

missä $0 \le q < p$ ja $0 \le f < 2p$. Tässä työssä käytetty testidata oli tuotettu seuraavilla arvoilla: $a = (2,2,1)$, $b = (0.5,0.5,0.5)$, $m = 0$, $n = (6,6,6)$. Tämä tuottaa muodoltaan kuusisakaraista tähteä muistuttavan kappaleen.

Liite B

Mallin tuottaminen power crust -algoritmilla

Algoritmin kehittäjien tekemä, vapaasti levitettävä toteutus löytyy osoitteesta **http://www.cs.utexas.edu/users/amenta/powercrust/welcome.html**. Sen kääntäminen Sun työasemilla onnistui ongelmitta ja se kääntynee hyvin useimmissa Unix ympäristöissä. Lisäksi koodista on saatavissa myös Windows ympäristössä toimiva versio osoitteesta http://www.cs.princeton.edu/~min/powercrust/. Paketti sisältää itseasiassa kolme eri ohjelmaa. Tärkein on varsinainen **powercrust**-ohjelma, jolla verkon muodostus tapahtuu. Lisäksi mukana on **orient** ja **simplify** -nimiset ohjelmat. Orient-ohjelman avulla powercrustilla tuotetun mallin pinnat voidaan kääntää loogisesti samaan suuntaan, sillä algoritmin suorituksen jälkeen osa niistä saattaa osoittaa pinnan sisäpuolelle ja osa ulkopuolelle. Simplify –ohjelmalla taas voidaan yksinkertaistaa powercrustin suorituksen aikana syntynyttä kappaleen keskiakselia. Tämä yksinkertaistettu keskiakseli voidaan antaa uudelleen syötteeksi powercrust-ohjelmalle.

Powercrustille syöte annetaan PTS tiedostona, joka on yksinkertainen ascii-muotoinen tapa kuvata pistejoukko. Siinä pisteiden koordinaatit on lueteltu tiedostossa aina yksi piste riviä kohden.

Mallin tuottaminen ja optimointi Hoppen algoritmeilla

Hugues Hoppen väitöskirjatyötään varten toteuttama koodi on saatavissa osoitteesta **http://research.microsoft.com/~hoppe/code.htm**. Paketti sisältää toteutukset sekä luvussa kolme esiteltyyn Hoppen verkonmuodostusalgoritmiin että luvussa neljä esiteltyyn energiafunktioihin perustuvaan verkon optimointialgoritmiin. Koodi on toteutettu C++-kielellä, ja sen kääntäminen Gnu C -kääntäjällä Sun työasemalle vaati hieman lisätyötä. Lähinnä ongelmia aiheuttivat muuttujien näkyvyysalueet, joita koskevat säännöt ovat ilmeisesti muuttuneet C++-standardissa sitten koodin kirjoittamisen. Mikäli kääntäjässä on jokin optio, jolla näkyvyyssääntöjä voi muokata, saattaa koodin saada kääntymään helpostikin. Muuten jokaista ongelmia aiheuttavaa kohtaa täytyy muokata käsin.

Paketin mukana tulee esimerkkitapauksia sisältävä hakemisto, jossa on **Reconstruct**-niminen skripti erään esimerkkitiedoston käsittelyyn. Tätä työtä varten tuotetut mallit

muodostettiin muokkaamalla tätä skriptiä sopivasti. Muokkaaminen tarkoittaa lähinnä tiedostonimen muuttamista ja sopivan näytetiheysparametrin antamista.

Itse tehdyt työkalut

Työn käytännönosuuden yhteydessä syntyi joukko pieniä apuohjelmia. Lähinnä niitä tarvittiin datan vaihtamiseksi formaatista toiseen, valmiiden mallien siivoamiseen ja metriikoiden mittaamiseen malleista. Mikäli joku tarvitsee tämänkaltaisia työkaluja, niin niiden lähdekoodi on saatavissa tekijältä. Ohjelmat on toteutettu ANSI C:llä ja paketti kääntyy sellaisenaan ainakin Sun työasemilla. Mukana ovat seuraavat työkalut:

Convert3d on työkalu 3d verkkojen muuttamiseksi formaatista toiseen. Ohjelma toimii kuten Unix maailmassa käytetty **convert-**niminen kuvaformaattien vaihtoon tarkoitettu ohjelma. Tällä hetkellä muunnokset onnistuvat vain Geomview OFF -formaatin ja Hoppen käyttämän m-formaatin välillä sekä rajoitetusti POV-Ray ohjelman POV-formaatin välillä. Uusia muunnoksia on kuitenkin helppo lisätä jälkeenpäin.

Meshinfo ilmoittaa OFF –formaatissa annetusta verkosta joukon metriikoita. Se ilmoittaa verkon kärkien, särmien ja kolmioiden lukumäärän sekä Eulerin karakteristikan. Lisäksi kerrotaan kolmioiden muotoa kuvaavia arvoja, sekä suurin löydetty kärjen asteluku.

Cleanoff on työkalu, jolla OFF-formaatissa annetusta verkosta voidaan siivota pois moninkertaiset kärjet ja kolmiot, joiden pinta-ala on nolla. Lopputuloksena annettu verkko on muutettu kolmioverkoksi, mikäli syöte sisälsi myös muita monikulmioita. Käytetty siivousalgoritmi on yksinkertainen ja hidas, ja sen tehokuutta voitaneen parantaa. Myöskin vaihtamalla kolmiointialgoritmia voidaan saada komioiden muoto geometrisesti paremmaksi, mikäli syötteessä on muitakin monikulmioita. Lähinnä tätä työkalua käytettiin power crust -toteutuksen antaman tuloksen siivoamiseen.

Pts2hoppe on työkalu, jolla PTS muodossa oleva pistejoukko voidaan muuttaa Hoppen algoritmin käyttämään muotoon.

Ptsclean käy läpi PTS muodossa annetun pistejoukon ja tarkistaa, ettei se sisällä moninkertaisia esiintymiä samasta pisteestä. Jos tällaisia löytyy, niin ne poistetaan.

Sphere on työkalu, jolla voidaan tuottaa satunnaista pallodataa. Se arpoo pallopinnalta halutun määrän pisteitä ja tallentaa ne PTS muotoiseen tiedostoon.

Metasphere on työkalu, joka tuottaa metapallodataa ja tallentaa sen PTS muotoiseen tiedostoon (ks. liite A).