KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Andrius Žukas

Trimačiai objektai: atvaizdavimo ir deformacijos algoritmai

Magistro darbas

Darbo vadovas doc. dr. Dalius Rubliauskas

KAUNAS, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Trimačiai objektai: atvaizdavimo ir deformacijos algoritmai

Informatika

MAGISTRO DARBAS

Magistrantas .....

Andrius Žukas IFM-2/1

2008 m. gegužės 26 d.

Vadovas_____

doc. dr. Dalius Rubliauskas

2008 m. gegužės 26 d.

Recenzentas_____

doc. dr. Eduardas Bareiša

2008 m. gegužės 26 d.

KAUNAS, 2008

Turinys

Projekto tikslai

Savo projekte aš siekiu tokių tikslų:

- Išanalizuoti realių pasaulio objektų nuskenavimo ir jų perkėlimo į virtualią erdvę galimybes. Susipažinti su jau esamomis idėjomis ir siūlymais kaip efektyviai atkurti nuskenuoto objekto paviršių.
- Pateikti savo algoritmus nuskenuoto trimačio objekto rekonstrukcijai. Siekiu sukurti Delaunay trianguliacijos principu veikianti metodą, į kurį įeina algoritmai nuskenuoto taškų debesies padalinimui į dalis ir jų perkėlimui į plokštumą, po trianguliacijos paviršiaus atstatymui naudojant trikampių tinklelius.
- Patikrinti sukurtų algoritmų teisingumą, įgyvendinant juos ir išbandant su realiais duomenimis.

1. ĮVADAS

Spartus technologijų tobulėjimas įneša daug naujovių į mūsų gyvenimą. Kartu su technologijomis žmogus turi žengti į priekį, kad išnaudotų visas jų teikiamas galimybes. Kartu, be žmogaus įsikišimo naujovės greitai užgestų.

Atsiradus ir vis labiau tobulėjant trimačių objektų skenavimo technologijoms įžvelgiamos labai plačios jų panaudojimo galimybės. Tai paskatino trimačių objektų rekonstrukcijos metodų atsiradimą. Jie vis labiau pakeičia tradicinius kompiuterinio modeliavimo metodus ir žavi naujom galimybėm greitai ir tiksliai gauti virtualius realaus pasaulio objektų modelius.

Turbūt greičiausiai trimatis skenavimas buvo pritaikytas kompiuterinių žaidimų pramonėje. Labai patogu žmones ir objektus iš tikrovės perkelti į žaidimų erdvę. Be to, rekonstrukcijos metodai taip pat leidžia ir redaguoti nuskenuotus, pridėti papildomų detalių, patobulinti geometrines linijas.

3D skeneriai gali skenuoti įvairius objektus. Priklausomai nuo jų veikimo principo, įmanoma rekonstruoti ir labai tolimus objektus. Čia atsiranda galimybės pritaikyti tokius prietaisus architektūrinių statinių skaitmenizavimui. Viena iš panaudojimo sričių galėtų būti istorinių, apgriuvusių, sugadintų realaus pasaulio objektų rekonstrukcija, kai nuskenuojamas nepilnas objektas, o pritaikius vizualizacijos ir deformacijos mechanizmus būtų galima atkurti trūkstamas detales.

Dar viena sritis, kur matomos didelės šių metodų perspektyvos yra ortopedinės technikos gamyba. Pagal nuskenuotas individualaus žmogaus kūno dalis galima sukurti tikslius protezus. Pavyzdžiui, pagal žmogaus pėdos modelį, būtų galima kurti kiekvienam žmogui individualiai pritaikytą avalynę, kuri būti patogi ir komfortiška.

Deja, kol kas patys trimačiai skeneriai dar nesugeba nuskenuoto realaus pasaulio objekto paversti į iškart tinkamą naudoti modelį. Todėl po skenavimo yra reikalingi metodai konvertuoti skenavimo įrenginių išduodamus rezultatus į toliau lengvai panaudojamus virtualius modelius. Konvertavimo mechanizmų yra daug, tačiau visi jie veikia panašiu principu. Nuskenuojamas realaus pasaulio objektas ir gaunamas taip vadinamas taškų debesis (taškų aibė, turinti formą, dydį, poziciją erdvėje). Paskui iš tokių debesų konstruojami paviršiai. Rekonstrukcijos metodų yra daug, veikiančių labai skirtingai, tačiau bene didžiausia problema yra paviršiaus atstatymo greitis.

Nors šiame darbe nagrinėjami bendri rekonstrukcijos metodai, tačiau didesnis dėmesys skiriamas metodams, tinkantiems ortopedinės technikos gamybai. Eksperimentams pasirinktas žmogaus pėdos modelis. Labiausiai akcentuojama ir ieškoma būdų paviršiaus rekonstrukcijos greičiui padidinti.

Analizės skyriuje pradedama apžvelgti literatūra susijusi su trimačiais modeliais, skenavimo būdais bei skenavimo įranga. Toliau apžvelgiami ir pagrindiniai rekonstrukcijos metodai, jų privalumai ir trūkumai.

2. ANALIZĖ

2.1. Modeliavimo apžvalga

Atliekant tyrimus, RE, pramonėje gaminant detales, kuriant naujus produktus vartotojams neapsieinama be skaitmeninių modelių. Skaitmeniniu modeliu vadinama realaus objekto virtuali kopija, kurią galima apdoroti naudojant kompiuterines sistemas (kompiuterius). Modelis gali būti aprašomas matematinėmis funkcijomis ir išraiškomis. Toks modeliavimo būdas nėra labai patogus: konstrukcijos gaunasi sudėtingos, sunkiai skaitomos, reikia specialių vizualizacijos priemonių. Kur kas labiau paplitę yra specialūs modeliavimo įrankiai (CAD – Computer Aided Design; CAM – Computer Aided Manufacturing). CAD sistemomis laikoma modeliavimo įrankių visuma, su kuriais galima kurti skaitmeninius objektus, atitinkančius realaus pasaulio objektus ir juos atvaizduoti. Jomis modeliai aprašomi geometrinių objektų visuma. Labiausiai paplitusios CAD sistemos: AutoCAD, Solid Works, CATIA, Maya, 3DS Max. Pastarosios dvi taip pat priskiriamos prie CAD sistemų, nors yra labiau skirtos realistiškiems vaizdams kurti, o ne brėžiniams braižyti bei tikslioms objektų kopijoms modeliuoti ir skaičiavimams atlikti.

2.2. Trimačių skenavimo įrankių apžvalga

Modeliavimas CAD sistemomis gali būti sudėtingas ir reikalauti daug laiko. Ypač atsiranda problemų kai susiduriama su sudėtingomis objekto formomis, tokiomis kaip žmogaus galva. Tobulėjant technologijoms atsiranda vis naujesnių skaitmeninių modelių kūrimo būdų. Šiame darbe realaus pasaulio objektams pervesti į virtualią erdvę naudojama trimačio skenavimo metodika. Tam yra naudojami specialūs išoriniai prietaisai. Įrenginių visuma, automatiškai fiksuojanti skanuojamo paviršiaus koordinates trejomis dimensijomis (3D koordinatėmis) sukuriant tam tikrą sistematišką struktūrą, yra laikoma trimačio skanavimo instrumentais. Tokiais prietaisais nesunkiai atkuriami ir sudėtingų formų paviršiai.

3D Skeneriai veikia labai panašiai kaip filmavimo kameros. Jie turi kūgio formos matymo lauką. Kameros savo matymo lauke fiksuoja ir saugo spalvinę informaciją, tuo tarpu trimačiai skeneriai vietoj jos saugo informaciją apie atstumą iki skanuojamo paviršiaus. Tokio

skenerio sukurtame atvaizde kiekvienas taškas aprašo atstumą iki jį atitinkančio taško paviršiuje. Vienas skanavimas sukuria pilną paviršiaus modelį. Keli skanavimai iš įvairių erdvės pozicijų reikalingi norint surinkti informaciją apie visas tiriamo objekto plokštumas. Tokių skenavimų metu gaunami paviršiaus duomenys yra registruojami, paskui suliejami į vieną visumą. Šitaip gaunas vientisas realaus pasaulio objekto modelis.

Pagal veikimo principą 3D skeneriai yra skirstomi į kontaktinius ir nekontaktinius. Šie savo ruožtu dar yra klasifikuojami į aktyviuosius ir pasyviuosius.

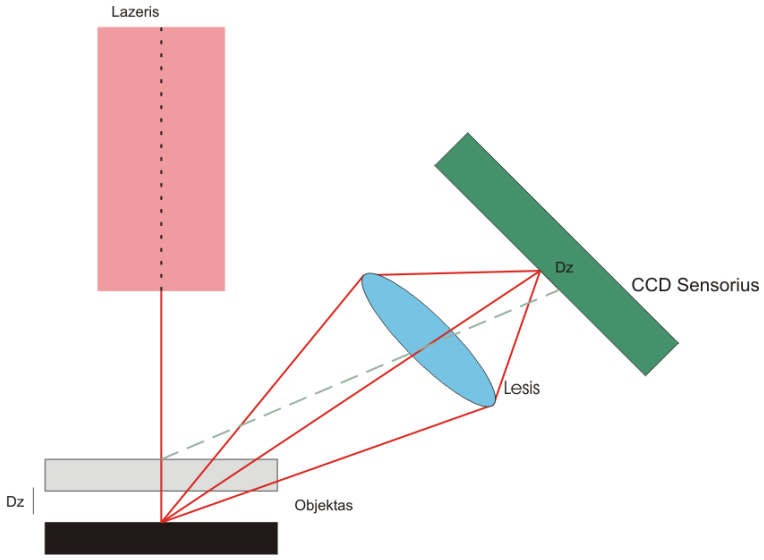
2.2.1. Kontaktiniai skeneriai

Kontaktiniai skeneriai analizuoja tiriamo objekto paviršių fizinio kontakto metu. Dažniai jie yra naudojami pramoninėje gamyboje. Tokių prietaisų privalumas – didelis tikslumas. Tačiau dėl veikimo kontaktiniu būdu tokie įrenginiai turi ir neigiamų savybių. Skenavimo metu gali būti deformuotas skenuojamas objektas, arba jis gali būti visai sugadintas. Toks veikimo principas yra sąlyginai lėtas, lyginant su kitokiomis skenavimo metodikomis.

2.2.2. Nekontaktiniai skeneriai

Nekontaktinių aktyviųjų skenerių veikimas pagrįstas radijo bangų spinduliavimu į tiriamąjį objektą ir atspindžių nuo jo aptikimu. Tinka įvairios prigimties radijo bangos, tiek šviesa, tiek ultragarsas ar rentgeno spinduliai. Labiausiai paplitę skeneriai naudojantys lazerio šviesą. Tokių prietaisų pagrindinis komponentas yra lazerio šviesos atspindžio detektorius. Jis matuoja per kiek laiko šviesa nukeliauja iki tiriamojo objekto ir atsispindėjusi grįžta atgal. Žinant šviesos greitį c, galima apskaičiuoti atstumą iki modeliuojamo objekto paviršiaus konkretaus vieno taško $atstumas = \frac{c \cdot t}{2}$. Skeneris vienu metu matuoja atstumą iki vieno paviršiaus taško, todėl visas paviršius peržiūrimas nuosekliai. Tokio tipo įrenginiai tinka skaitmenizuoti objektus, esančius dideliu atstumu nuo paties įtaiso. Todėl jie yra pritaikomi skenuojant dideles struktūras, pavyzdžiui pastatus. Veikdamas dideliais atstumais aparatas praranda tikslumą. Sunkiai atpažįstami skenuojamo objekto kampai, nes šiose srityse padidėja šviesos išsklaidymas.

Priešingas savybes turi taip vadinamas trianguliacijos skeneris. Jis taip pat klasifikuojamas kaip nekontaktinis aktyvusis skeneris. Jo veikimo atstumas nėra didelis,



1 pav. Trianguliacijos skeneris

tačiau tikslumas pasiekiamas iki dešimčių mikrometrų. Trianguliacijos skeneriai yra sudaryti iš lazerio, skaitmeninės kameros ir lęšių sistemos. Priklausomai kurioje vietoje lazerio spindulys atsimuša į skenuojamą paviršių, atsispindėjęs jis fiksuojamas skirtingose kameros matymo lauko vietose. Tokia metodika vadinama trianguliacija, nes lazeris, jo spindulio taškas ant skenuojamo paviršiaus ir kamera sudaro trikampį (žr. 1 pav.). Atstumas tarp lazerio ir kameros yra žinomas. Trikampio kampas, kuriame yra lazeris taip pat žinomas. Trikampio kampas, kuriame yra kamera gali būti nustatytas, pagal tai, kurioje vietoje yra lazerio spindulio taškelis kameros matymo lauke. Tokios informacijos pilnai pakanka nustatyti trikampio dydžiui ir formai, bei fiksuoti skanuojamo objekto paviršiaus tašką erdvėje. Dažnai vietoj lazerio taško naudojama ant skenuojamo objekto paviršiaus krentanti lazerio juosta. Taip paspartinamas skenavimo procesas.

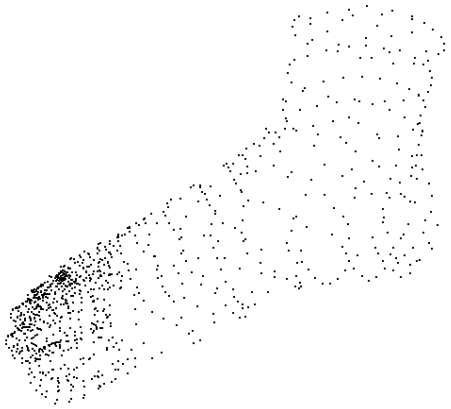
Įvertinus 3D skenerių savybes, jų veikimo principus šiame darbe naudojamas trianguliacijos skeneris, nes labiausiai atitinka nagrinėjamos problemos pobūdį. Toliau šiame darbe naudojami minėtu skeneriu gauti duomenys.

2.3. Trimačiu skeneriu gautų rezultatų analizė

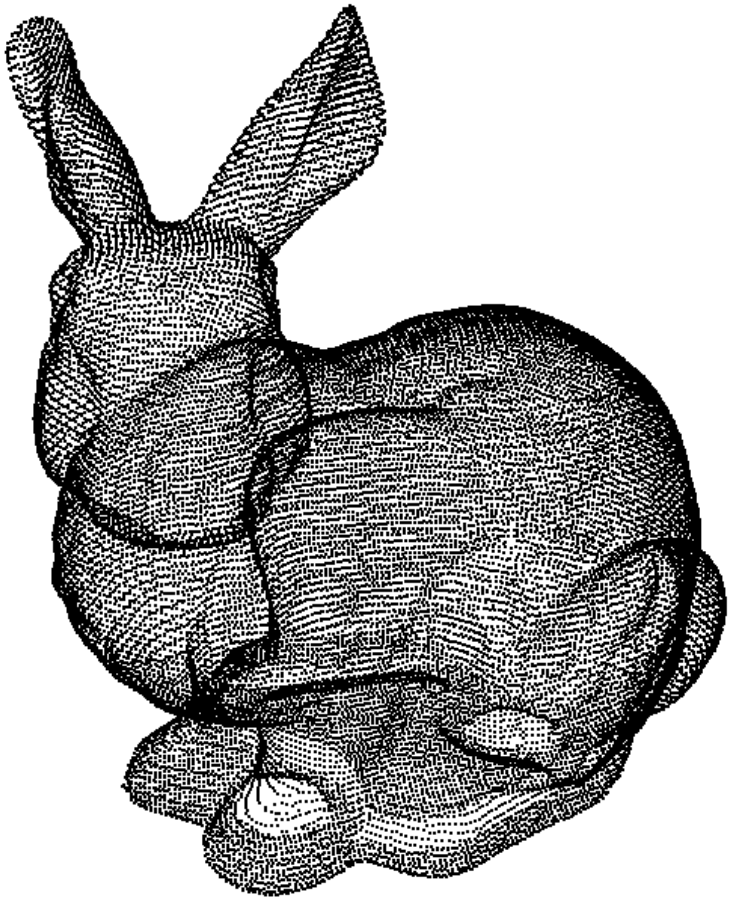
3D skeneriu gautas realaus pasaulio objekto modelis yra sudarytas ne iš matematinių funkcijų ar kreivių visumos. Jis yra aprašomas taškų aibe. Tokia taškų aibė vadinama „Taškų debesiu**“**

Taškas p yra žymimas kaip $p = (x, y, z)$, kur x, y ir z yra koordinatės Dekarto koordinačių sistemoje. Aš naudojamu P pažymėti visų galimų taškų p aibei, $p \in P \subseteq \mathbb{R}^3$. Tuomet Taškų debesį P' apibrėžiame kaip visų galimų taškų aibės poaibį, kuris aprašo skenuojamo objekto geometrinę formą ir padėtį erdvėje, $p \in P' \subseteq P \subseteq \mathbb{R}^3$. d žymi Euklido atstumą tarp dviejų taškų p_1 ir p_2: $d = \|p_1 - p_2\|$, kur $p_1, p_2 \in P$.

Trimačiai skeneriai gali generuoti tolygiuosius arba netolygiuosius taškų debesis. Jei Dekarto atstumas d tarp kiekvieno gretimo taškų debesies taško p_n ir p_{n+1}, $p_n \in P'$ yra vienodas arba beveik vienodas su paklaida ε tai laikoma, kad taškų debesis yra tolygusis (žr. 2 pav.), $d_1 \pm \varepsilon = d_2$, kur d_1 yra atstumas tarp taškų p_1 ir p_2, o d_2 yra atstumas tarp taškų p_1 ir p_3, čia p_1, p_2 ir $p_3 \in P'$. Jeigu Dekarto atstumas d tarp gretimų taškų debesies taškų p_n ir p_{n+1}, $p_n \in P'$ yra nevienodas tai toks taškų debesis vadinamas netolygiuoju (žr. 3 pav. (1)). Taškų p tankumas taškų debesyje P' priklauso nuo trimačio skenerio, kuriuo buvo skenuojamas realaus pasaulio objektas, taip priklauso nuo to, kiek kartų buvo atliekamas skenavimas ir kitų parametrų. Šiame darbe nagrinėjami abiejų tipų taškų debesys.



2 pav. Netolygusis taškų debesis. Pėdos modelis



3 pav. Tolygusis taškų debesis. Triušio modelis

2.4. Taškų debesų panaudojimo analizė

Trimačiu skeneriu nuskanavus realaus pasaulio objektą gautą taškų debesį tiesiogiai panaudoti sunku. Norint atlikti deformacijas, ar kokius nors skaičiavimus (tūrio, aukščio) su trimačiu modelių reikia, kad jis būtų aprašytas geometriniais objektais (tiesėmis, kreivėmis, splainais, trikampiais, kvadratais ir pan.).

2.4.1. NURBS splainų panaudojimo analizė

Labai dažnai modelių paviršiai aprašomi splainais (1). Dėl savo lankstumo ir gebėjimo formuoti bet kokią taisyklingą ar netaisyklingą figūrą labiausiai išpopuliarėjo NURBS splainai (2; 3). NURBS (Non-Uniform Rational B-Spline) – neunifikuoto racionaliojo B-Splaino pagrindas yra *Bezier* kreivė. B-Splainas yra sudarytas iš kelių Bezier segmentų su tam

tikru tęstinumo laipsniu, lokalizuojant kontrolinių taškų poveikį kreivei bei pridedant svorius. Šio tipo kreivės naudojamos dėl patogumo skaičiuoti kompiuteriu, stabilumo slankaus kablelio klaidų atžvilgiu, mažų atminties sąnaudų, reikalingų konstrukcijos saugojimui ir galimybės įgyti bet kokią formą.

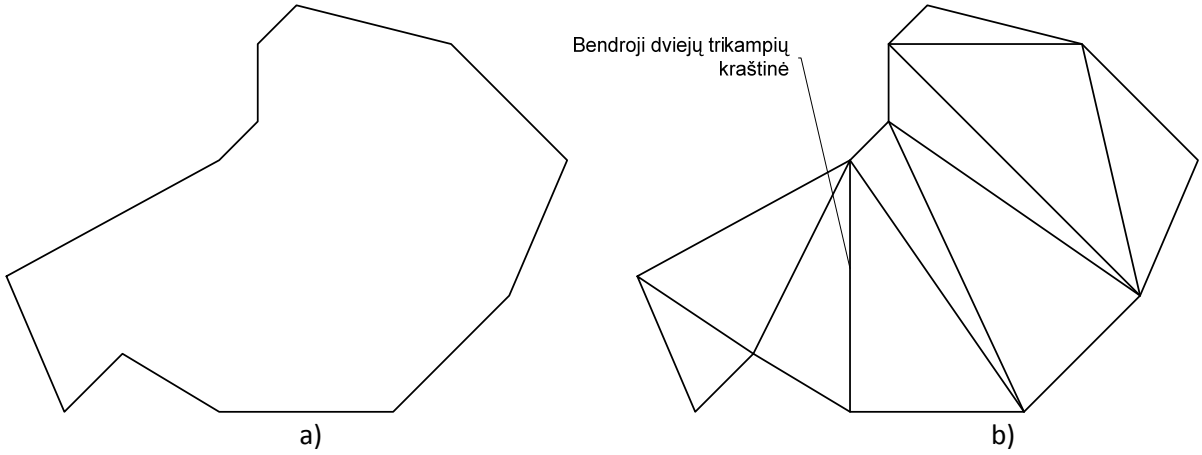
Kadangi NURBS splainai gerai aprašo praktiškai bet kokį paviršių, atsirado būtinybė šį interpoliavimo metodą pritaikyti ne tik naujų modelių kūrimui, bet ir RE (4), kuomet trimatis modelis gaunamas iš taškų debesies. (4) siūlo metodą, leidžiantį atvirkštinės inžinerijos metodu iš taškų debesies rekonstruoti objekto paviršių ir iškarto gauti CAD modelį. Šio metodo veikimo principas pagrįstas paviršiaus išlenkimų skaičiavimu, jo ribų nustatymu ir NURBS paviršiaus pritaikymu. Norint sukurti tokį tinkantį paviršių reikia žinoti kiekvieno paviršiaus taško poziciją erdvėje. Taškų pozicijos parametrai suskaičiuojami naudojant parametrizavimo metodus. Pradžioje gaunamas bazinis paviršius, kuris sekančiose iteracijose yra patikslinamas. Priklausomai nuo atkuriamo modelio sudėtingumo gali tekti sujungti kelis paviršius su skirtingais parametrais sujungimo linijoje. Todėl galimi atkurto paviršiaus iškraipymai.

2.4.2. Trianguliacijos metodų analizė

Kur kas labiau žinoma ir naudojama metodika paviršiams rekonstruoti iš taškų aibės yra vadinama trianguliacija (5). Trianguliacija yra plokštumos ar paviršiaus daugiakampių sudalinimas į trikampių aibę. Paprastai yra taikomi apribojimai, tokie, kad kiekvieną trikampio kraštinę dalinasi du gretimi trikampiai (žiūrėti 4 pav.). Buvo įrodyta, kad kiekvienas paviršius turi trianguliaciją, tačiau jai sukurti gali prireikti begalinio skaičiaus trikampių (6). Paviršiaus trianguliacija, turinti baigtinį skaičių trikampių, vadinama kompaktiška. Wickham-Jones (1994) pristato $O(n^3)$ sudėtingumo algoritmą trianguliacijoms skaičiuoti (7). 1998 O'Rourke papildo pasiūlytą metodą ir sudėtingumą sumažina iki $O(n^2)$ (8). Garey (8) (1978) pasiūlo naują algoritmiškai nesudėtingą metodą ($O(n \ln \ln n)$) paviršiams trianguliuoti, kuris daugelį metų laikytas optimaliu. Tačiau vėliau buvo pasiūlyta ir $O(n \lg \lg n)$ sudėtingumo algoritmų, o 1991 Chazelle (9) netikėtai parodo, kad pasirinktas paprastas daugiakampis gali būti trianguliuotas per $O(n)$ laiką.Tačiau pasak kai kurių tyrėjų šį algoritmą yra beviltiška įgyvendinti.

Taškų debesies trianguliacijos problema sprendžiama labai panašiai. Iš pradžių apskaičiuojamas iškilasis briaunainis, gaubiantis visus taškų debesies taškus. Paskui jo sienos

dalinamos į trikampius pagal apskaičiuotus parametrus (pvz. paviršiaus išgaubtumas) iki reikiamo lygio, kol pasiekiamas paviršiaus tikslumas, atitinkantis realaus pasaulio objektą.



4 pav. Daugiakampio trianguliacija.
a) pradinis daugiakampis b) daugiakampis, sudalintas į trikampius

Paviršiaus rekonstrukcijos metodai skirstomi į tris kategorijas. Šiame skyriuje apžvelgsime keletą pagrindinių problemos sprendimo požiūrių.

„Skaptavimo“ metodai

„Skaptavimo“ (Sculpting-based) metoduose pirmiausia yra apskaičiuojama taškų debesies Delaunay trianguliacija (10). Paskui išrenkami tie trikampiai, kurie turi įtakos objekto formai. Boissonnat (11) taiko 3D DT, gaunamas taškų debesies iškilasis briaunainis. Jeigu ne visi debesies taškai patenka ant briaunainio sienų, vadinasi kažkurie tetraedrai turi būti pašalinti iš trianguliacijos. Iteracijos kartojamos tol, kol visi taškai lieka gulėti ant briaunainio sienų. Tuo pačiu visą laiką turi būti išsaugoma briaunainio struktūra. Crust algoritme apskaičiuojama Voronojaus diagrama pradinei taškų aibei. Pagal Voronojaus viršūnes atrenkami toliausiai esantys poliai. Jie sujungiamo su Voronojaus diagramos viršūnių aibe. Toliau skaičiuojama Delaunay trianguliacija. Galiausiai atrenkami trikampiai, kurių viršūnės patenka į taškų debesies aibę. Lyginant su kitais šis algoritmas yra sąlyginai lėtas. Labiausiai laiką eikvojantys veiksmai jame yra 3D Voronojaus diagramos skaičiavimas ir 3D Delaunay trianguliacijos skaičiavimas. Pažymėtina, kad sujungus polius su taškų debesim maksimalus jų skaičius yra $3n$. Taigi visas Crust algoritmas įvykdomas per $O(n^2) + O(9n^2)$.

„Kontūrų klijavimo“ algoritmai

Hoppe (12) algoritmas yra tipinis *„kontūrų kopijavimo“ (contour-tracing)* metodų pavyzdys. Nagrinėjant problemą šiuo požiūriu erdvė yra suskirstoma į kubus ar ketursienius, įvertinamos jų viršūnių skaliarinės funkcijos. Kiekvieno iš šių elementų viršūnės yra laikomos apytikslėmis paviršių nusakančių parametrų reikšmėmis. Šios grupės algoritmai daugiau aproksimuoja nei interpoliuoja norimą paviršių. Kartu yra nufiltruojami žemo dažnio signalai. Tai toleruotina jei turime taškų debesį su triukšmais. Kitu atveju informacija tiesiog prarandama.

„Srities audimo“ metodai

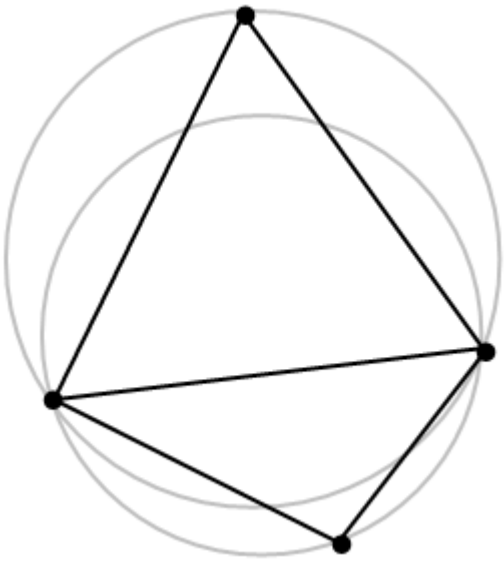
„Srities augimo“ (Region-growing) iteraciniuose metoduose paviršius rekonstruojamas pradedant nuo vieno atrinktojo pradinio trikampio. Palaipsniui prie dalinio paviršiaus prijungiami nauji trikampiai kol suformuojamas pilnas modelis. Trikampių kraštinės, prie kurių jungiami naujai apskaičiuoti trikampiai, vadinamos aktyviomis kraštinėmis. Pagrindinė šių metodų problema yra tokio naujo taško parinkimas iš taškų debesies, kuris su aktyvia kraštine sudarytų naują trikampį daliniame paviršiuje. Šios grupės BPA (13) algoritme naudojama apie aktyvią kraštinę besisukanti sfera su vartotojo nurodytu spinduliu. Kai sfera paliečia greta aktyvios kraštinės esantį tašką, jis yra pasirenkamas ir suformuojamas naujas trikampis. Huang ir Menq (14) savo pasiūlytame metode ant plokštumos, einančios per aktyviajai kraštinei gretimą trikampį, projektuoja k artimiausių aktyviosios kraštines galams taškų. Iš k taškų pagal trumpiausio ilgio kriterijų atrenkamas vienas taškas, kuris suformuoja naują trikampį su aktyviąja kraštine.

Daug paviršiaus rekonstrukcijos iš taškų debesies metodų remiasi 3D Delaunay trianguliacijos skaičiavimu. Pagrindinė jų problema interpoliavimo greitis. Geriausiu atveju skaičiavimams reikia $O(n^2)$ laiko. (15) straipsnis paremtas Crust algoritmu. Autoriai siūlo du sprendimo būdus: padidinti 3D Voronojaus diagramos skaičiavimo efektyvumą arba sumažinti taškų debesyje esantį taškų skaičių. Kadangi Voronojaus diagramos ir Delaunay trianguliacijos studijuojamos nuo to laiko kai buvo pristatytos 1936 metais, problema bandoma spręsti antruoju būdu, mažinamas taškų debesyje esančių taškų skaičius. Pastebėta įdomi savybė. Rekonstruojamame trimačiame modelyje ne visur reikalingi vienodo dydžio trikampiai paviršiui aprašyti. Mažą išgaubtumą turintys paviršiai gerai interpoliuojami didesniais trikampiais, tuo tarpu, staigiai keičiantiems kryptį reikia mažesnių trikampių.

Aprašant didesnį trikampį užtenka mažesnio skaičiaus taškų. Nagrinėjant žmogaus galvos modelį skruostų srityje galėtų būti sumažintas taškų skaičius, o nosies, ausų ir lupų srityje reikia išsaugoti visus taškų debesies taškus. Skaičiuojant 3D Voronojaus diagramą su mažesnėmis taškų aibėmis reikia mažiau laiko. Tačiau dažnai neišsaugoma kai kurių svarbių atkuriamo modelio detalių. Be to, susiduriama su problema nustatant sritis, kurias nėra būtina aprašyti labai mažais trikampiais.

2.5. Delaunay trianguliacija

Tarkime taškas t žymimas kaip $t = (x, y)$ kur x ir y yra koordinatės Euklido koordinačių sistemoje. Tuomet T yra plokštumoje esančių taškų t aibė, $t \in T \subseteq \mathbb{R}^2$.
Tokia trianguliacija $DT(T)$ vadinama Delaunay trianguliacija, jei nei vienas taškas $t \in T$ nėra viduje apibrėžtinio apskritimo (16) apie bet kurį trikampį iš trianguliacijos $DT(T)$ (žiūrėti 4 pav.). DT maksimizuoja trianguliacijoje esančių trikampių mažiausius kampus iš visų kampų.
Remiantis DT apibrėžimu teigiama, kad apie trikampį apibrėžtas apskritimas per tris trikampio viršūnes yra tuščias, jei jame nėra jokių kitų viršūnių, nei tų, per kurias apibrėžtas apskritimas (kitų trikampių viršūnės gali būti tik ant apskritimo perimetro, o ne apskritimo viduje).



5 pav. Delaunay Trianguliacijai būtina sąlyga (17)

Iš būtinos DT būtinos sąlygos galima teigti, kad trikampių tinklelis yra Delaunay trianguliacija, jeigu apie kiekvieną trikampį apibrėžtas apskritimas yra tuščias. Analogiškai DT galima apibendrinti ir trimačiam atvejui, tik tuomet vietoj apibrėžtinio apskritimo naudojama sfera.

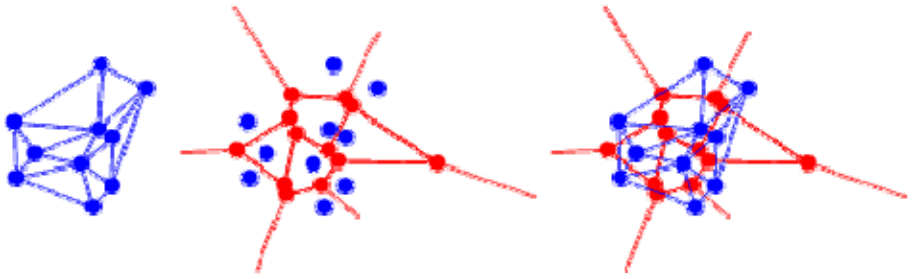
Ryšys su Voronojaus diagramomis

Taškų aibės T Delaunay trianguliacija atitinka Voronojaus diagramos taškų aibei T dualiajam grafui. Žinant VD galima gauti Delaunay trianguliaciją (žiūrėti 5 pav.).

2.5.1. Delaunay Trianguliacijos savybės

Tarkime n yra taškų $t \in T$ skaičius, o dim yra dimensijų skaičius

- Trianguliacijoje esančių visų ketursienių sąjunga suformuoja iškiląjį briaunainį (žr. 6 pav.)
- DT sudaro daugiausiai $O(n^{\lceil \frac{d}{2} \rceil})$ ketursienių.



a) **b)** **c)**

6 pav. a) Delaunay Trianguliacija b) Voronojaus Diagrama c) Ryšys tarp DT ir VD (18)

- Kai $dim = 2$, jeigu egzistuoja b skaičius viršūnių ant iškiliojo briaunainio, tai bet kokia taškų trianguliacija daugiausiai turės $2n - 2 - b$ trikampių, plius vieną išorinę sieną.
- Delaunay Trianguliacija maksimizuoja mažiausią kampą. Lyginant su bet kokia kita taškų trianguliacija, mažiausias kampas DT yra bent jau tokio dydžio, kaip ir mažiausias kampas bet kurioje kitoje trianguliacijoje. DT nebūtinai minimizuoja didžiausius trikampių kampus.
- Apie bet kurį Delaunay trikampį apibrėžtas apskritimas savo viduje neturi jokių kitų taškų debesies taškų.

- Jei per du taškus einantis apibrėžtinis apskritimas neturi daugiau nei vieno taško savo viduje, tai linijos segmentas tarp šitų dviejų taškų yra DT briauna.
- Delaunay Trianguliacija $d-$ dimensinėje erdvėje yra iškilojo briaunainio taškų projekcijos ant $(d+1)$ – dimensinėje aplinkoje esančio paraboloido.

2.6. Analizės išvados

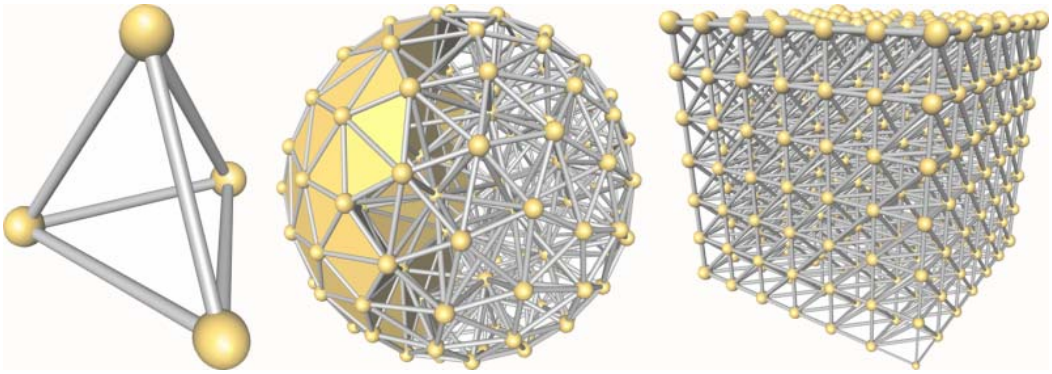
Aukščiau paminėtos literatūros analizė atskleidė, kad trianguliacijos naudojimas paviršiams rekonstruoti iš taškų aibės turi privalumų lyginant su kitais rekonstravimo būdais. Splaininės konstrukcijos gerai interpoliuoja paviršius, tačiau norint toliau redaguoti rekonstruotą objektą įterpti reikiamą skaičių kontrolinių taškų, paviršių suskaidyti į tinklelius. Kad virtualus modelis kuo labiau atitiktų realaus pasaulio objektą, darant NURBS kreivių priderinimą reikia iškart atlikti daug deformacijų. Vėliau gali iškilti problemų dėl per didelio kontrolinių taškų skaičiaus ir neefektyviai išnaudojamos atminties.

Trianguliacija šiek tiek aproksimuoja rekonstruojamo objekto paviršius, tačiau vėliau taikant glotninimo funkcijas atstatomi ir labai tikslus paviršiai. Trikampių tinklelis iškart atkartoja taškų debesies kontūrus, todėl nebereikia taikyti papildomų deformacijos algoritmų ir iškart yra gaunama reikiama realaus pasaulio objekto forma ir dydis. Trianguliacijos metoduose atminties išeikvojimo problema taip pat yra aktuali. Tiksliai nuskanuoti realaus pasaulio objektai aprašomi taškų debesimis, kuriuose yra milijonai taškų. Jungiant taškus į trikampių viršūnes gaunamos papildomos duomenų matricos su jungiamų viršūnių indeksais. Visa tai didina atminties naudojimo sudėtingumą. Tačiau pritaikius taškų debesies segmentaciją nesunkiai galima išskirti paviršiaus sritis, kuriose taškų tankumas nėra labai svarbus, ir tose srityse sumažinti jų skaičių iki minimumo. Labai svarbi yra Delaunay Trianguliacija. Jos savybės leidžia sukonstruoti optimalius trikampius. Naudojant netolygiuosius taškų debesis modelio srityse, kur reikalingas detalesnis paviršius, automatiškai gaunamas tankesnis tinklelis. Praktiškai pritaikant trianguliaciją galima rinktis iš didelės metodų ir algoritmų gausos.

Nors jau labai daug tyrimų atlikta trianguliacijos srityje, tačiau manau, kad šie paviršiaus rekonstrukcijos metodai yra perspektyvūs, ir galima juose atlikti pakeitimų, padidintų veikimo efektyvumą. Savo darbe aš pristatau naują metodą greitesnei 3D trianguliacijai atlikti. Jis yra paremtas Delaunay Trianguliacijos skaičiavimu. DT pasirinkta dėl savybių gausos, taip pat yra daug jau sukurtų algoritmų DT apskaičiavimui, kuriuos galima panaudoti savo metoduose.

3. SIŪLOMO PAVIRŠIAUS REKONSTRUKCIJOS METODO KONCEPCIJA

Literatūros analizės metu nustatyta, kad skaičiuojant 3D Delaunay Trianguliaciją reikia nemažiau $O(n^2)$ laiko. Tokios trianguliacijos rezultatas – gaunami iškilieji briaunainiai. Paviršiaus dydžiui ir formai aprašyti užtenka atskirų briaunainių sienų (žr. 7 pav.). Interpoliuojant taškų debesį daugiausiai laiko sunaudojama trimačių briaunainių, tenkinančių Delaunay sąlygas, paieškai. Be to, suradus tokius briaunainius reikia išsaugoti tik tas sienas, kurios formuoja rekonstruojamo modelio paviršių.



7 pav. Trimatis Delaunay Trianguliacijos atvejis. Paviršiui aprašyti naudojamos ne visos briaunainio sienos (19)

Aš siūlau sumažinti skaičiavimų sudėtingumą taikant dvimatį Delaunay trianguliacijos atvejį. DT tinklelį reikia apskaičiuoti taškams, išdėstytiems plokštumoje. Trianguliacijoje gaunami tik reikalingi trikampiai, aprašantys paviršių. Galiausiai rekonstruojamo modelio taškų debesį reikia „apvilkti“ plokštumoje gautu trianguliacijos tinkleliu. „Sweepline“, „Divide and Conquer“, kai kurie augimo („Incremental“) algoritmai apskaičiuoja DT plokštumoje per daugiausiai $O(n \log n)$ laiką. Reikia patikrinti, ar įmanoma efektyviai trimatį taškų debesį transformuoti į dvimatį, neprarandant rekonstruojamo objekto formą ir dydį nusakančių parametrų. Taip pat reikia įsitikinti, ar yra būdas gautą dvimatį trikampių tinklelį „apvilkti“ ant trimačio taškų debesies.

3.1. Pagrindiniai siūlomo metodo žingsniai

Pagal pagrindinius atliekamus veiksmus algoritmą greitai paviršių rekonstrukcijai galima suskirstyti į tokius etapus:

- Taškų debesies taškų perkėlimas į plokštumą
    - Projektavimo plokštumos pasirinkimas
    - Dalijimo plokštumos radimas
    - Taškų debesies P' dalinimas į poaibius P_1' ir P_2'
    - Taškų perskirstymas plokštumoje
- Dvimatės Delaunay Trianguliacijos apskaičiavimas
- Taškų debesies „apvilkimas“ dvimačiu Delaunay tinkleliu
- Viso modelio trimačio modelio atvaizdavimas

4. SIŪLOMAS PAVIRŠIAUS REKONSTRUKCIJOS METODAS

Tarkime turime trimačiu skeneriu gautą taškų debesį P'. Jis gali būti tolygusis arba netolygusis. Kadangi Delaunay Trianguliaciją apskaičiuojama plokštumoje, kiekvienas taškų debesies taškas $p \in P'$ projektuojamas į vieną iš koordinačių plokštumų: Oxy, Oxz ar Oyz.

4.1. Taškų debesies taškų perkėlimas į plokštumą

4.1.1. Projektavimo plokštumos pasirinkimas

Siekiant gauti gerą taškų debesies P' projekciją plokštumoje reikia tinkamai pasirinkti vieną iš koordinačių plokštumų, į kurią bus projektuojami visi taškai p. Taško p projekciją plokštumoje pažymėkime $pp_{kp} = (x, y)$, kur x ir y yra koordinatės Dekarto koordinačių sistemoje; $kp = Oxy, Oxz, Oyz$ (pažymima koordinačių plokštuma). Tuomet PP_{kp} nusako viso taškų debesies projekciją plokštumoje, $pp_{kp} \in PP_{kp}$, čia $kp = Oxy, Oxz, Oyz$. Taškų debesies projekcijos plokštuma pasirenkama taip, kad projektavimo algoritmas geriausiai pasiskirstytų kiekvieno taško projekcijas pp su atitinkamais atstumais tarp jų. Labiausiai tinkama projekcijos plokštuma laikoma ta, kurioje taškų debesies projekcijos PP iškilojo daugiakampio plotas yra didžiausias. Taškų debesies P' projekcijų PP_{Oxy}, PP_{Oxz} ir PP_{Oyz} iškilieji daugiakampiai ID apskaičiuojami pagal (1) išraišką.

$$ID_{kp} = convhull\left(PP_{kp}\right), \text{ čia } kp = Oxy, Oxz \text{ ir } Oyz \qquad (1)$$

Iškiliojo daugiakampio ID skaičiavimui naudojamas Qhull (20) metodas. Šiuo metodu apskaičiuojami ir iškiliųjų daugiakampių ID plotai S:

$$S_{kp} = S_{convhull}(ID_{kp}), \text{ čia } kp = Oxy, Oxz \text{ ir } Oyz \qquad (2)$$

Pagal (2) išraišką suskaičiuojame taškų debesies P' projekcijų PP plotus kiekvienoje iš trijų plokštumų. Tolimesniems skaičiavimams atlikti pasirenkama plokštuma, kurioje iškiliojo

daugiakampio ID plotas S yra didžiausias, nes taip užtikrinamas geriausias trimačio taškų debesies kiekvieno taško $p \in P'$ pasiskirstymas dvimatėje plokštumoje. Projekcijos plokštuma pasirenkama vykdant funkciją $ProjekcijosPlokstuma(PP)$ (žr. 1 algoritmą).

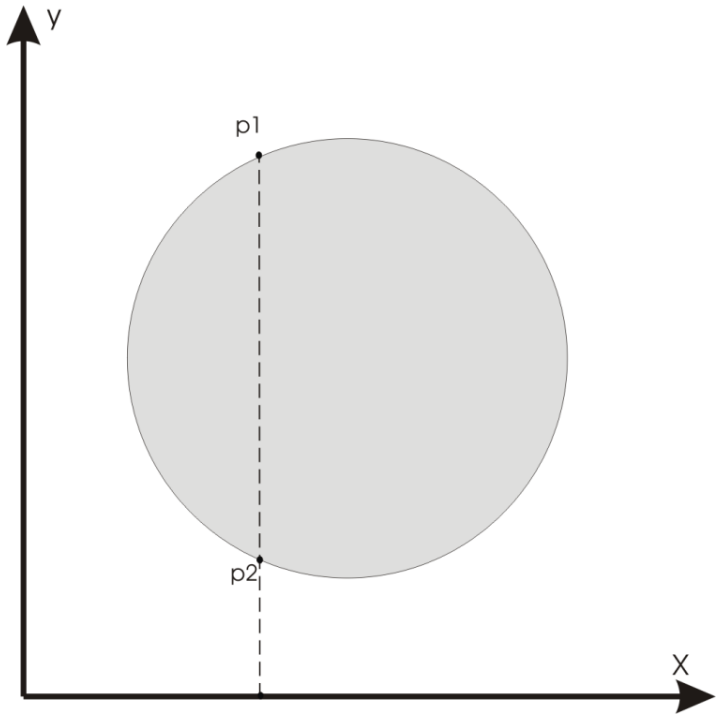
```
Duomenys: PP
Rezultatas: ProjP
1 ProjekcijosPlokstuma(PP) begin
2   kp = {Oxy, Oxz, Oyz};
3   while kp ≠ ∅ do
4       ID_kp = convhull(PP_kp);
5       S_kp = S_convhull(ID_kp);
6   ProjP = max(S_kp);
7   return ProjP;
8 end
```

1 algoritmas. Projekcijos plokštuma

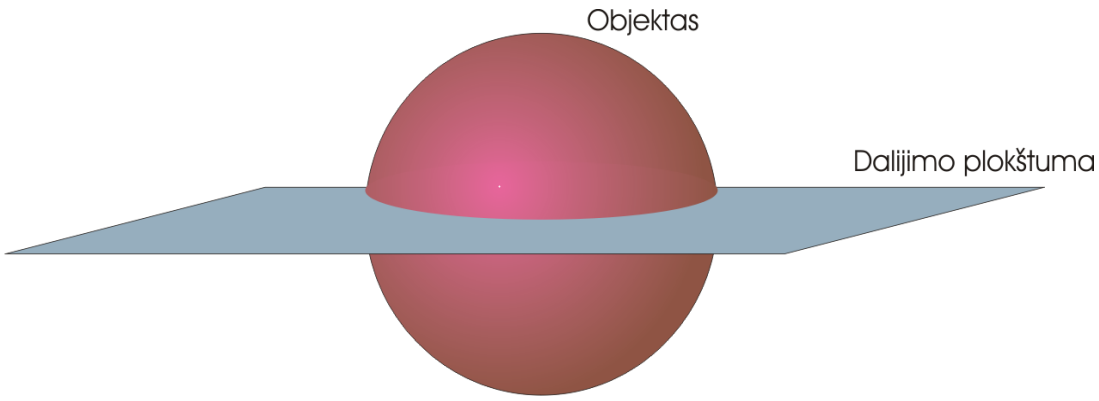
Funkcija $ProjekcijosPlokstuma(PP)$ ima taškų debesies P' projekcijas kiekvienoje koordinačių plokštumoje, kiekvienos iteracijos metu suskaičiuoja iškiliojo daugiakampio plotą visose koordinačių plokštumose. Gražina koordinačių plokštumą, kurioje minėtas plotas yra didžiausias.

4.1.2. Dalijimo plokštumos radimas

Sekantis algoritmo žingsnis – dalijimo plokštumo radimas. Šios plokštumos funkcija yra trimatį taškų debesį P' padalinti į dvi dalis gaunat du mažesnius taškų debesies modelius P'_1, $P'_2 \subseteq P'$. Panagrinėkime atvejį, kai dar nepadalinto taškų debesies P' visi taškai p yra projektuojami ant aukščiau surastos projektavimo plokštumos. Kadangi tai atvejis, kai trimatės erdvės taškai projektuojami į dvimatę erdvę, gaunami taškų persidengimai (žr. 8 pav. Taškas $p1$ perdengia tašką $p2$).



8 pav. Taškų persidengimas projekcijoje

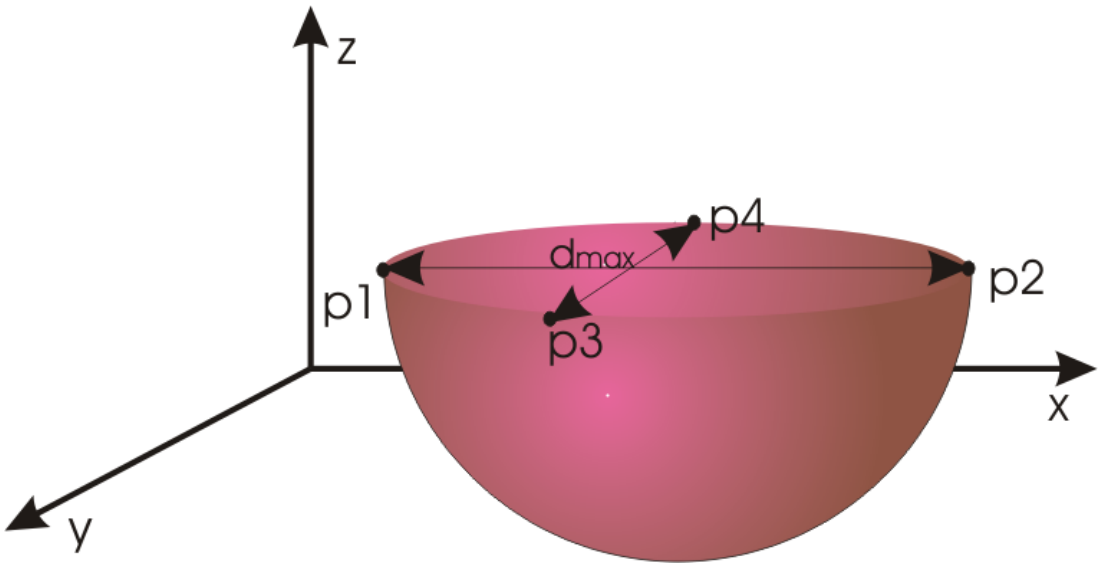


9 pav. Objektą kertanti dalijimo plokštuma

Projektuojant 3D tašką $p = (x, y, z)$ į plokštumą Oxy gaunama jo projekcija $pp = (x, y)$. Nepadalinus taškų debesies į dvi dalis, gaunami taškų, esančių pačiame debesies viršuje ir pačioje debesies apačioje persidengimai arba nykstamai maži suartėjimai. Dalijimo plokštumos suformuoti du mažesni taškų debesies P' poaibiai P'_1 ir P'_2 padeda išvengti galimų taškų persidengimų(žr. 9 pav.).

Tarkime taškų debesį P' projektuosime į koordinačių plokštumą Oxy. Tuomet optimaliausia, kad dalijimo plokštuma kirstų P' tokiame aukštyje z, kuriame atstumas d tarp labiausiai

nutolusių taškų p_1 ir p_2 koordinačių ašies x atžvilgiu ir taškų p_3 ir p_4 koordinačių ašies y atžvilgiu yra didžiausias (žr. 10 pav.).



10 pav. Dalijimo plokštumos taškų pasirinkimas

Dalijimo plokštuma formuojama per taškus per taškus p_1, p_2, p_3 ir p_4. Funkcija $DalijimoPlokstuma(P')$ gražina dalijimo plokštumą DP (žr. 2 algoritmą).

Duomenys: P', PP_{kp}
Rezultatas: DP
1 $DalijimoPlokstuma(P', PP_{kp})$ ***begin***
2 $kp \in \{Oxy, Oxz, Oyz\}$;
3 $T = \boldsymbol{sort}(PP_{kp}, x)$;
4 $T1 = \boldsymbol{sort}(PP_{kp}, y)$;
5 $DPT = \{P'(T_{pirmas}); P'(T_{paskutinis}); P'(T1_{pirmas}); P'(T1_{paskutinis})\}$;
6 $DP = plane(DPT)$;
7 ***return*** DP;
8 ***end***

2 algoritmas. Dalijimo plokštumos radimas

2 algoritme taškų debesies projekcijos PP_{kp} taškai išrūšiuojami pradžioj pagal x koordinačių ašį (3 algoritmo žingsnis), paskui pagal y koordinačių ašį (4 algoritmo žingsnis). Kadangi ieškoma labiausiai nutolusių taškų pagal x koordinačių ašį, tai išrūšiuotame taškų pp masyve didžiausias atstumas d bus tarp pirmojo ir paskutiniojo taško. 5 algoritmo žingsnyje

suformuojama taškų aibė, per kuriuos eina dalinamoji plokštuma. Projekcijos taškas turi tik x ir y koordinates, todėl 5 žingsnyje trečioji z koordinatė priskiriama iš taškų debesies P'. 6 žingsnyje per rastus taškus suformuojama dalinamoji plokštuma.

4.1.3. Taškų debesies P' dalinimas į poaibius P_1' ir P_2'

Norint išvengti anksčiau minėto taškų persidengimo projekcijose, į projekcijos plokštumą reikia projektuoti ne visą taškų debesį P', o atskirus jo poaibius. Tam tikslui buvo sukonstruota dalijimo plokštuma. Poaibiu P_1' žymėsime visus taškų debesies P' taškus, esančius aukščiau dalijimo plokštumos, o poaibiu P_2' žymėsime taškus, esančius žemiau dalijimo plokštumos.

Funkcija $SudarytiPoaibius$, pateikta 3 algoritme, įgyvendina taškų debesies P' dalinimą į poaibius P_1' ir P_2'.

Duomenys: P', DP
Rezultatas: P_1', P_2'
1 $SudarytiPoaibius(P', DP)$ **begin**
2 **for each** $p \in P'$ **do**
3 $if\ (p_z \geq DP_z)$;
4 $P_1' = add(p)$;
5 $else$
6 $P_2' = add(p)$;
7 **return** P_1', P_2';
8 **end**

3 algoritmas. Poaibių sudarymas

Poaibių sudarymo funkcijoje (3 algoritmas) peržiūrimas kiekvienas taškų debesies P' taškas p, ir pagal jo z koordinatę nustatoma, jei jis yra virš dalinimo plokštumos DP (3 algoritmo žingsnis), tai šis taškas patenką į poaibį P_1', priešingu atveju – į poaibį P_2'.

4.1.4. Taškų perskirstymas plokštumoje

Praeitame etape taškų debesis P' buvo išskaidytas į dvi mažesnes taškų aibes P_1' ir P_2'. Nuo šiol nagrinėsime tik vieną aibę P_1', nes su antrąja aibe atliekami analogiški veiksmai. Taškus perskirstyti plokštumoje reikia kad gautųsi teisinga 2D Delaunay trianguliacija. Vien tik atlikus trimačių taškų projekcijas plokštumoje, kai kuriose vietose gauname taškų sutankėjimus. Taškų išlyginimui siūlau atlikti tokius veiksmus: apskaičiuoti atstumą d tarp taškų p_1 ir p_2 trimatėje erdvėje ir su gautu atstumu atidėti taškų projekcijas pp plokštumoje. To užtenka gauti teisingą Delaunay trianguliaciją plokštumoje.

Funkcijos $PerskirstytiTaskus(P_1')$ (žr. 4 algoritmą) įvedimo duomenys yra taškų debesies P' poaibis P_1'. Funkcija grąžina perskirstytą taškų poaibį PT. Visi poaibio P_1' taškai išrūšiuojami pagal z koordinatę. 3 algoritmo žingsnyje randamas taip vadinamas centrinis taškas cp. Tai yra aukščiausiai poaibyje P_1' esantis taškas. Nuo jo skaičiuojamas atstumas iki kiekvieno kito poaibio taško p. cp apibrėžkime kaip $cp = (x, y, z)$, o bet kurį kitą tašką $p = (x_1, y_1, z_1)$, $cp, p \in P_1'$. Tuomet atstumas apskaičiuojamas pagal (3) išraišką.

$$d = \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} \tag{3}$$

Perskirstant taškus svarbu išsaugoti taškų debesies geometrinę formą. Todėl taškas $p \in P_1'$ pastumiamas tam tikra kryptimi, kurią gauname įvykdę funkciją $direction(cp, p)$. Kryptis nustatoma remiantis tiesės lygtimi ((4) išraiška) ir taškais cp ir p.

$$y = mx + b \tag{4}$$

Kadangi Delaunay trianguliacija atliekama plokštumoje reikia visus taškų debesies poaibio P_1' taškus p suprojektuoti į projekcijos plokštumą PP. 9 žingsnyje funkcija $move(pp, dir, d)$ perkelia taško projekciją pp apskaičiuota kryptimi dir, atstumu d nuo centrinio taško cp.

Duomenys: $P_1^{'}$
Rezultatas: PT
1 $Perskirstyti\ taškus(P_1^{'})$ ***begin***
2 $sorted = \boldsymbol{sort}(P_1^{'}, z);$
3 $cp = sorted(1);$
4 $PT = PT \cup cp;$
5 ***for each*** $p \in P'$ ***do***
6 $d = \|cp - p\|;$
7 $dir = \boldsymbol{direction}(cp, p);$
8 $pp = \boldsymbol{project}(p);$
9 $nloc = \boldsymbol{move}(pp, dir, d);$
10 $PT = PT \cup nloc;$
11 ***return*** $PT;$
12 ***end***

4 algoritmas. Taškų perskirstymas

4.2. Dvimatės Delaunay Trianguliacijos apskaičiavimas

Dvimatės Delaunay trianguliacijos apskaičiavimui naudojamas Qhull (20) metodas. Nors blogiausiu atveju jo veikimo laikas yra $O(n^2)$, tačiau jis pasirinktas dėl papildomų savybių gausos. Vietoj Qhull algoritmo galima naudoti bet kurį kitą 2D Delaunay trianguliacijos skaičiavimo būdą, veikiantį greičiu nei blogiausiu atveju per $O(n^2)$ (pvz.: „Sweep line, „Divide And Conquer“). Funkcija (5) išraiškoje gražina aibę taškų indeksų Tri, kuriuos jungiant gaunamas Delaunay tinklelis.

$$Tri = delaunay(PT) \qquad (5)$$

4.3. Taškų debesies „apvilkimas“ dvimačiu Delaunay tinkleliu

Apskaičiavę dvimatę Delaunay trianguliaciją žinome taškų pp indeksus, kuriuos reikia sujungti, kad gautume trikampių tinklelį. Į funkciją, atvaizduojančią Delaunay triangulaiciją, paduodami du parametrai: trianguliacijos taškų indeksų aibė Tri ir projekcijos taškų aibė PP,

su kuriais buvo atliekama trianguliacija. Vietoj projekcijos taškų aibės PP į funkciją $Apvilkimas(Tri, P_1')$ (žr. 5 algoritmą) paduosime taškų debesies aibės P' poaibį P_1'. Šio poaibio taškai $p = (x, y, z)$ turi aukščio parametrą – koordinatę z, todėl Delaunay tinklelis atvaizduojamas ne plokštumoje, o erdvėje.

Duomenys: $Tri, P_1^{'}$
Rezultatas: $mesh$
1 $Apvilkimas(Tri, P_1^{'})$ ***begin***
2 ***for each*** $p \in P_1'$ ***do***
3 $t = connect(p, Tri)$;
4 $mesh = mesh + t$;
5 ***return*** $mesh$;
6 ***end***

5 algoritmas. Taškų debesies apvilkimas dvimačiu Delaunay tinkleliu.

Funkcija $Apvilkimas$ 3 žingsnyje sujungia kiekvieną tašką p su kitais poaibio P_1' taškais, pagal trianguliacijos taškų indeksų masyvą Tri. Algoritmas grąžina taškų debesies P' poaibį P_1', „apvilktą“ Delaunay tinkleliu, parametre $mesh$. Tai yra aukščiau dalijimo plokštumo DP esantis poaibis. Kitas poaibis P_2' apskaičiuojamas analogiškai, į funkcijas vietoj P_1' kaip parametrą paduodant P_2'.

4.4. Viso modelio trimačio modelio atvaizdavimas

Atvaizduojant taškų debesį P', padalinta į du poaibius P_1' ir P_2' reikia, kad jie būtų sujungti ir gautume vientisą trimatį objektą. Taškų debesies dalijimo funkcijoje $SudarytiPoaibius$ (žr. 3 algoritmą) tiek į poaibį P_1', tiek ir į poaibį P_2', patenka taškų debesies P' taškai, esantys dalijimo plokštumoje DP. Įvykdžius trianguliacijos funkciją, pateiktą (5) išraiškoje, gauti Delaunay trikampių tinkleliai šiek tiek persidengia, todėl atvaizdavus tokią trianguliaciją, gaunamas vientisas, trimatis realaus pasaulio objekto modelis. Atvaizdavimas atliekamas įvykdžius funkciją $draw(mesh)$ (žr. (6) išraišką). Kaip parametras į šią funkciją paduodamas trikampiais sujungtas taškų debesis $mesh$.

$$draw(mesh) \tag{6}$$

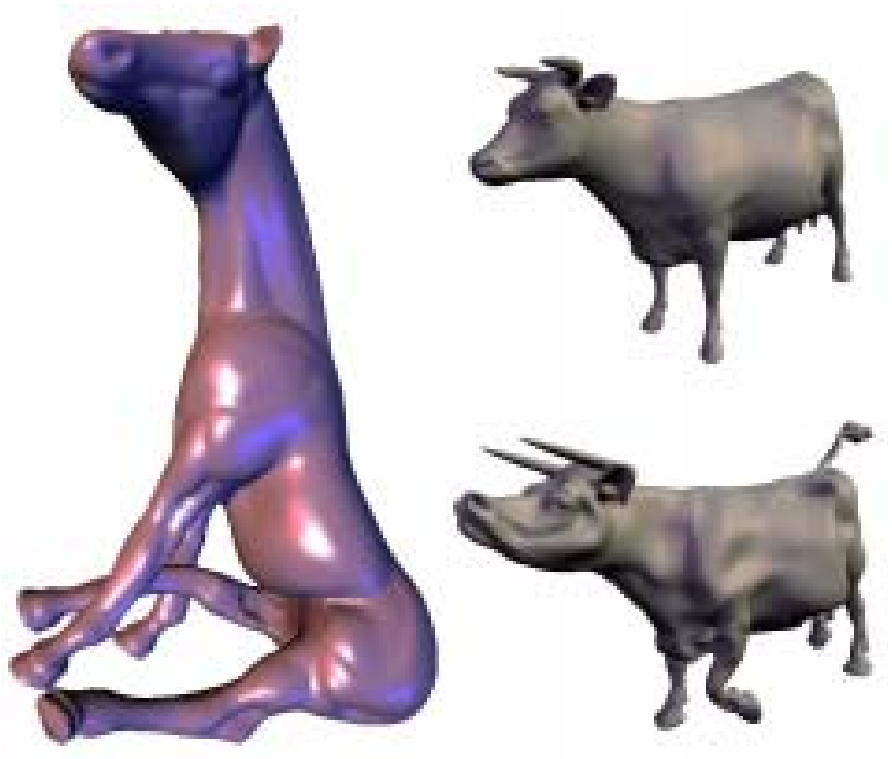
5. ATKURTO PAVIRŠIAUS DEFORMACIJOS

Siekiant padidinti rekonstrukcijos algoritmų praktinį pritaikymą reikia išplėsti trimačių atkurtų modelių panaudojimo galimybes. Iškarto po paviršiaus atstatymo iš nuskenuoto taškų debesies modelį galima naudoti vizualizacijai, formos ir dydžio įvertinimui, dažnai tokie modeliai naudojami žaidimų industrijoje. Galimybės išplečiamos, jei virtualų objektą įmanoma deformuoti. Praktikoje rekonstruotiems paviršiams pritaikomi įvairūs deformacijų šablonai: iteraciniai; panaudojant deformacijų kreives; deformuojant atskirus geometrinius paviršiaus objektus (trikampių dydžio keitimas, pasukimas, veidrodinio atspindžio naudojimas); naudojant bangelių transformacijas; paviršiaus redagavimas keliais lygiais (angl. Multiresolution Editing) (21). Pastarasis labiausiai nagrinėjamas literatūroje.

Šiame darbe nagrinėjamos teorinės galimybės pritaikyti deformacijos algoritmus, kadangi pagrindinis projekto tikslas ir didžiausias dėmesys skiriamas modelio rekonstrukcijai iš nuskenuoto taškų debesies. Dėl plataus taikymo pavyzdžių ir unikalių savybių siūlau naudoti ME algoritmus. Jie pagrįsti paviršiaus „padalijimo“ į mažesnius vienetus (angl. Subdivision) koncepcija. „Padalijimas“ pradedamas nuo turimo paviršiaus. Taikant „skaidymo“ schemas (angl. Refinement schemes) paviršiaus trikampių tinklelis skaidomas į mažesnius trikampius kol pasiekiamas reikiamas lygis. „Skaidymo“ schemos skirstomos į dvi grupes: aproksimuojančias ir interpoliuojančias. Po padalijimo taikant aproksimuojančią schemą gaunamas ne toks tikslus paviršius. ME algoritmuose po trikampių viršūnių perkėlimo į naujas pozicijas taikomas „padalinimas“ ir tokių būdų išsaugomas paviršiaus detalių tikslumas.

Šiame darbe siūlau naudoti I. Guskov ir kitų autorių sukurtu paviršiaus redagavimo keliais lygiais metodu (22). Praktiškai jis gerai tinka todėl, kad veikia su netaisyklingais taškų debesimis. Skenuojant realaus pasaulio objektus dažniausiai ir gaunami tokie taškų debesys. Siekiant išsaugoti tikslumą, taškų debesyje apie objekto detales (pvz.: tiriant žmogaus galvą, apie akis, nosį, burną) susikaupia didesnis taškų skaičius. Metode, atliekant „padalijimą“, taikoma interpoliuojanti suskaidymo schema. Vadinasi, po virtualaus modelio deformacijos tikslumas neprarandamas (žr. 11 pav.). Tai dar vienas privalumas pasirenkant šį metodą savo projekte.

11 Paveiksle parodytas deformacijos pavyzdys.



11 pav. virtualaus modelio deformacija (22)

Atvirkštinės rekonstrukcijos metodą panaudojus ortopedinių gaminių gamyboje galima gaminti individualią avalynę pagal vartojo užsakymą. Nuskanavus užsakovo pėdą, aukščiau minėtą deformaciją galima būtų pritaikyti modeliavimo procese, patikrinant, kaip kuriama avalynė tinka ant sulenktos pėdos darant žingsnį, arba kaip pėda deformuojasi veikiama tam tikros jėgos.

6. SISTEMOS PROTOTIPAS

Norėdamas patikrinti savo idėjas įgyvendinau šiame darbe aprašytus algoritmus. Kadangi projekte buvo siekiama sukurti bendrus metodus nuskenuoto realaus pasaulio objekto paviršiaus atkūrimui ir modifikacijai, kuriuos vėliau būtų galima panaudoti specifinėse programinėse įrangose, prototipas neturi vartojo sąsajos, algoritmai veikia su keliais duomenų pavyzdžiais. Įgyvendinimui naudojau *„Matlab“* paketą. Duomenys saugomi tekstiniame faile. Duomenys pačiame faile pateikiami $x\,y\,z$ formatu, kur x, y ir z yra koordinatės Dekarto koordinačių sistemoje, atitinkančios taško $p = (x, y, z)$ koordinates, $p \in P'$. Tarpusavyje koordinatės duomenų faile yra atskirtos tarpais. Rezultatų išvedimui naudojamos standartinės *„Matlab“* funkcijos.

Svarbiausios kodo dalys pateikiamos Priede Nr.1.

7. EKSPERIMENTAI

Šiame skyriuje aprašomi mano atlikti eksperimentai bei pateikti jų rezultatai. Eksperimentuose nagrinėju 4 skyriuje pateiktų algoritmų veikimą su trimačiais taškų debesimis. Bandymams naudojami tiek tolygieji, tiek ir netolygieji taškų debesys. Eksperimentais siekiama nustatyti:

- Kokiu greičiu rekonstruojamas paviršius. Palyginti su kitais žinomais rekonstrukcijos algoritmais.
- Kaip auga paviršiaus rekonstrukcijos greitis didėjant taškų skaičiui taškų debesyje.
- Palyginti kaip skiriasi siūlomo metodo vykdymo laikai, jei vietoj 2D Delaunay trianguliacijos naudojama trimatė Delaunay trianguliaciją.
- Įvertinti rekonstruoto paviršiaus kokybę.

Visų eksperimentų metu naudojama techninė įranga, atitinkanti šiuos reikalavimus:

- Mikroprocesoriaus dažnis (angl. CPU frequency) – 1500 MHz, Pentium.
- Operatyviosios atminties (angl. RAM) kiekis – 1024 MB (DDR2 tipo)

7.1. Rekonstrukcijos greičio įvertinimas. Palyginimas su kitais algoritmais

Šis bandymas buvo atliekamas penkis kartus su kiekvienu taškų debesies modeliu. Vidutinė algoritmų veikimo laiko reikšmė apskaičiuota pagal formulę:

$$\bar{t} = \frac{1}{N}\sum_{i=1}^{N} band_i \qquad (7)$$

čia $\bar{t}$ yra vidutinis skaičiavimo laikas; N bandymų skaičius

Rezultatai pateikiami 1 lentelėje.

1 lentelė. Paviršiaus rekonstrukcijos greitis siūlomais metodais

		Kiekvieno bandymo laikas t, s					
Taškų debesies modelis	Taškų skaičius taškų debesyje	1	2	3	4	5	Vidurkis, $\bar{t}$
Sfera	252	0,115	0,102	0,115	0,109	0,116	0,111
Pėda	1876	0,595	0,414	0,416	0.423	0,437	0,457
Debesis1	34834	2,209	2,247	2,121	2,101	2,240	2,184
Debesis2	123748	7,545	7,566	7,572	7,667	7,518	7,574

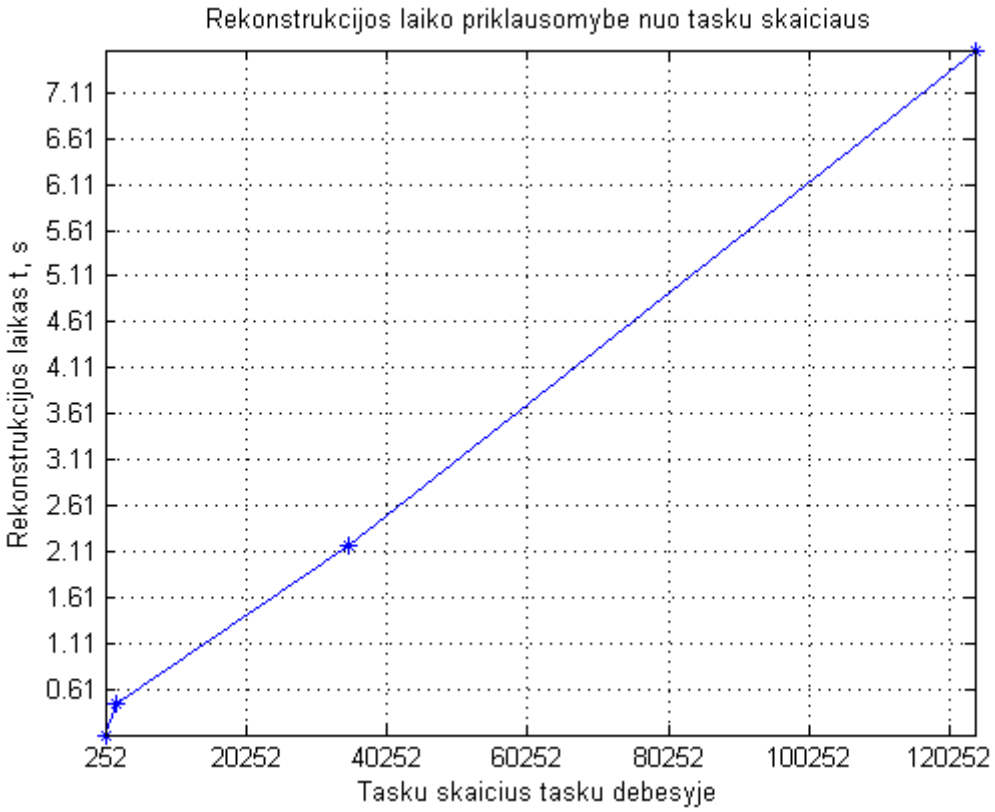
Palyginsime siūlomo metodo vykdymo laiką su kitais paviršiaus rekonstrukcijos algoritmais (žr. 2 lentelę).

2 lentelė. Algoritmų palyginimas

	Kiti algoritmai		Mano siūlomas metodas	
Algoritmas	Taškų skaičius	Vykdymo laikas t,s	Taškų skaičius	Vykdymo laikas t,s
IPD (23)	37888	85	34834	2,2
Hoppe (12)	1000	19	1876	0,5
Clustering (24)	134345	82,9	123748	7,6
DBRG (25)	34834	4,4	34834	2,2

2 lentelėje sulyginamų algoritmų taškų skaičius taškų debesyse nėra vienodas, nes nebuvo galimybės gauti kitų autorių naudotų pradinių duomenų. Tačiau ir iš tokio metodų sulyginimo atskleidžiami mano siūlomo metodo privalumai. Greičiausiai veikiantis algoritmas DBRG vis tiek pagal vykdymo laiką 2 kartus atsilieka nuo šiame darbe nagrinėjamų algoritmų.

7.2. Rekonstrukcijos greičio priklausomybė nuo taškų skaičiaus.



12 pav. Paviršiaus rekonstrukcijos laiko priklausomybė nuo taškų skaičiaus

Iš 12 pav. galima nustatyti, kad rekonstrukcijos laikas tiesiškai priklauso nuo taškų skaičiaus taškų debesyje.

7.3. Dvimatės ir trimatės Delaunay trianguliacijos palyginimas

Šis eksperimentas parodo, ar apsimoka naudoti 2D Delaunay trianguliaciją kartu su papildomais taškų debesies apdorojimo metodais ar geriau pasinaudoti trimačiu DT metodu. Tiek dvimačiu, tiek ir trimačiu atveju Delaunay trianguliacijai yra naudojamas Qhull metodas (20).

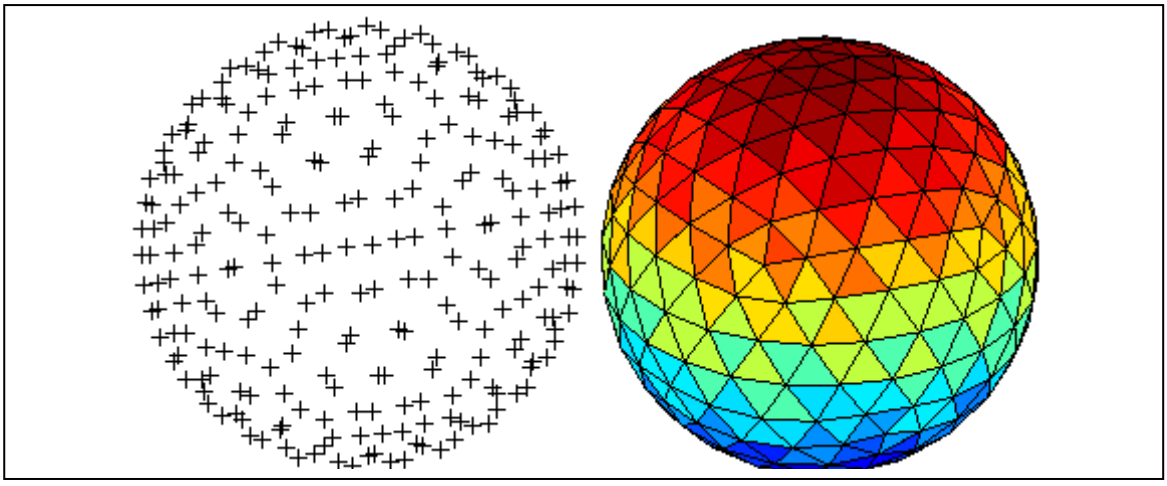
3 lentelė. Delaunay trianguliacijų palyginimas

		Bandymo laikas t, s	
Taškų debesies modelis	Taškų skaičius taškų debesyje	2D DT	3D DT
Sfera	252	0,111	0,090

Pėda	1876	0,457	0,379
Debesis1	34834	2,184	6,356
Debesis2	123748	7,574	25,433

Iš 3 lentelės galima spręsi, kad taškų debesims, turintiems labai nedidelį taškų skaičių tinka ir trimatė Delaunay trianguliacija, paviršiaus rekonstrukcijos laikai beveik vienodi. Tačiau taškų skaičiui stipriai išaugus geriau pasirinkti šiame darbe siūlomą metodą.

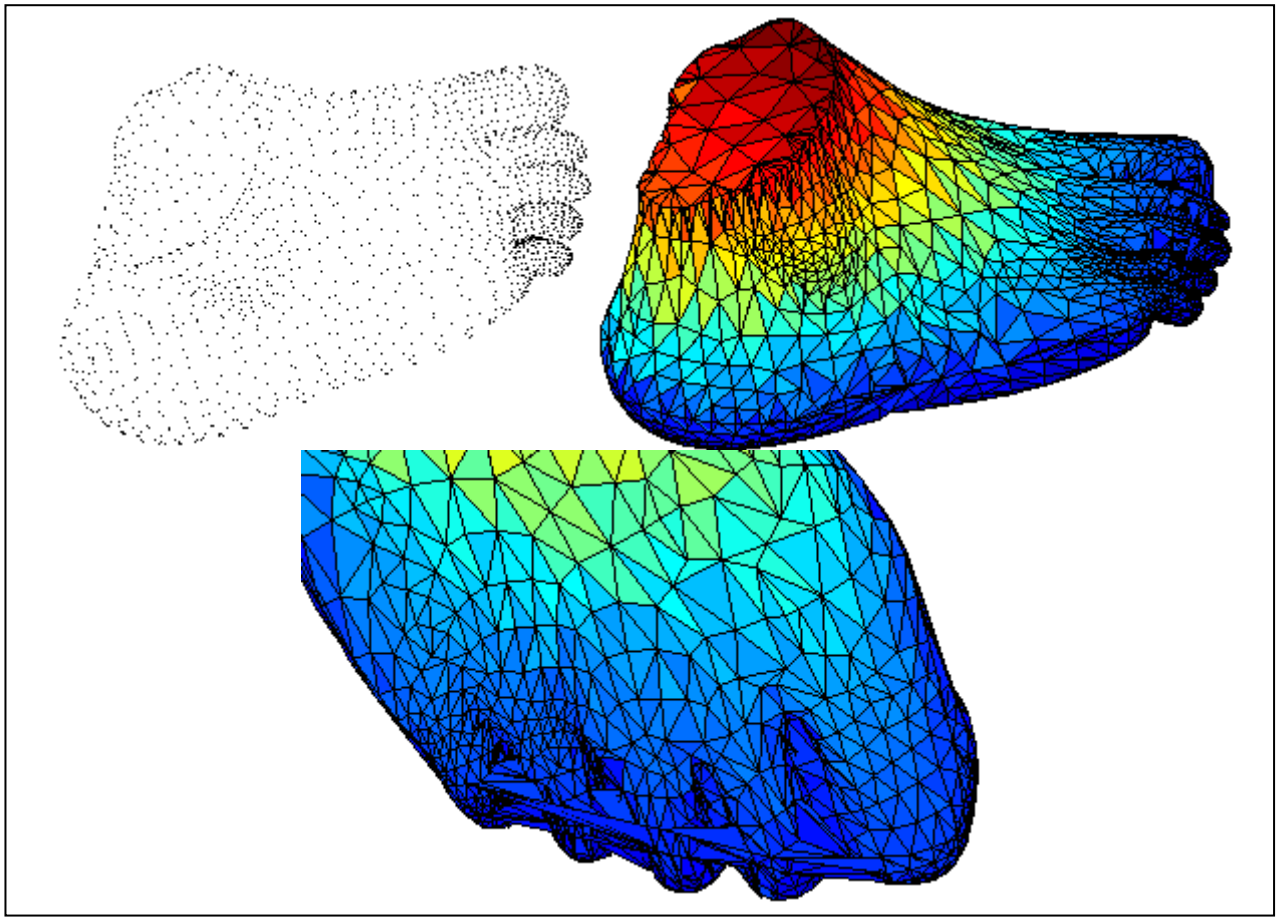
7.4. Rekonstruoto paviršiaus kokybės įvertinimas



13 pav. Sfera. Kairėje taškų debesis. Dešinėje atkurtas paviršius

13 pav. pateikiamas rekonstruotas sferos paviršius naudojant mano sukurtu metodu. Taškų debesis atvaizduojamas „+“ ženklu, kad geriau matytųsi. Tai yra tolygusis taškų debesis. Kairėje sfera atvaizduojama naudojant trikampių tinklelį. Matome, kad paviršius atkurtas kokybiškai, jis sudarytas iš taisyklingų trikampių.

14 pav. pateikiamas žmogaus pėdos rekonstrukcijos pavyzdys. Taškų debesis yra netolygusis. Atkurtas paviršius yra kokybiškas. Čiurnos srityje matomas trikampių tinklelio sutankėjimas,



14 pav. Žmogaus pėda. Kairėje taškų debesis. Dešinėje iš jo atkurtas paviršius. Apačioje pirštų fragmentas

vadinasi metodas išsaugo modelio detales. Apatiniame paveiksle pateikiamas pirštų fragmentas. Tarpai tarp jų užpildyti tinkleliu. Taip yra dėl to, kad skaičiuojant Delaunay trianguliaciją yra naudojamas iškilasis daugiakampis. Siūlomas šios problemos sprendimo būdas yra pateiktas Priede Nr.2.

8. IŠVADOS

Spartus technologijų tobulėjimas, naujų skenavimo įrenginių atsiradimas atveria kelius naujiems modeliavimo būdams plėtotis. Atvirkštinė inžinerija tampa vis svarbesne taikymų sritimi, kurioje nuskenuoti realaus pasaulio objektai yra gerai ir greitai rekonstruojami, bei pritaikomi tolesniuose modeliavimo procesuose.

Jau sukurtų metodų ir algoritmų paviršiaus rekonstrukcijai analizė parodė, kad egzistuoja didelė algoritmų veikimo laiko problema. Buvo užsibrėžtas tikslas šią algoritmų vykdymo laiko problemą sumažinti.

Aš sukūriau algoritmus vykdyti paviršiaus rekonstrukciją Delaunay trianguliacijos pagrindu. Mano sumodeliuoti algoritmai suranda taškų debesies dalijimo plokštumą, iš trimatės erdvės perkelia taškus į plokštumą ir, po plokštumoje atliktos Delaunay trianguliacijos, yra atstatomas trimatis virtualus modelis.

Atlikau eksperimentus savo idėjoms ir algoritmams ištestuoti, iš kurių matosi, kad mano metodika, pasitvirtino, pavyko greičiau įvykdyti paviršiaus rekonstrukcija lyginant su kitais, anksčiau skelbtais metodais.

9. ŽODYNĖLIS

4 lentelė. Santrumpų žodynas

Sąvoka	Paaiškinimas
CAD	Modeliavimas kompiuterio pagalba (angl. Computer Aided Design)
CAM	Kompiuteriu aprašyta gamyba (angl. Computer Aided Manufacturing)
NURBS	Neunifikuotas Racionalusis B-Splainas (Non-Uniform Rational B-Spline)
DT	Delaunay trainguliacija
VD	Voronojaus diagrama
DP	Dalijimo plokštuma
PP	Projektavimo plokštuma
RE	Atvirkštinė inžinerija (angl. Reverse enginering)
PT	Perskirstytų taškų poaibis
ProjP	Projekcijos plokštuma

10. LITERATŪRA

1. 3D Models, Plugins, Textures, and more at Turbo Squid. [Tinkle] [Cituota: 2008 m. Kovo 1 d.] http://www.turbosquid.com/.

2. **Boor, Carl de.** *A Practical Guide to Splines.* s.l. : Springer, 1978. ISBN: 978-0-387-95366-3.

3. **Jankauskas, Kęstutis ir Tautvydas, Perminas.** *Gaminio modelio analizė ir koregavimas pritaikant CAM sistemoms.* Kaunas : Kauno Technologijos Universitetas, 2006.

4. **Piegl, Les ir Tiller, Wayne.** *The NURBS Book.* s.l. : Springer, 1997. ISBN: 978-3-540-61545-3.

5. *Reverse engineering modelling of free-form surfaces from point.* **Kruth, J.-P ir Kerstens, A.** s.l. : Elsevier, April 1998 m., T. 76. S0924-0136(97)00341-5.

6. **Bern, Marshall ir Eppstein, David.** *MESH GENERATION AND OPTIMAL TRIANGULATION.* California : World Scientific, 1992.

7. **Francis, G. K ir Weeks, J. R.** Conway's ZIP Proof. *American Math.* 1999 m., 106.

8. **Wickham-Jones, T.** *Mathematica Graphics: Techniques and Applications.* New York : Springer-Verlag, 1994.

9. **Garey, M. R, et al.** *Triangulating a Simple Polygon.* s.l. : Inform. Process. Lett, 1978.

10. **Chazelle, B.** *Triangulating a Simple Polygon in Linear Time.* s.l. : Disc. Comput. Geom., 1991.

11. *Two algorithms for constructing a Delaunay triangulation.* **Lee, D. T ir Schachter, B. J.** s.l. : Springer Netherlands, June, 1980 m., T. 9. 0885-7458.

12. *Geometric structures for three-dimensional shape representation.* **Boissonnat, Jean-Daniel.** 4, New York : ACM, October 1984 m., T. 3. 0730-0301.

13. *Surface Reconstruction from Unorganized Points.* **Hoppe, Hugues, et al.** 2, s.l. : Computer Graphics, July 1992 m., T. 26.

14. *The ball-pivoting algorithm for surface reconstruction.* **Bernardini, F, et al.** 4, s.l. : Computer Graphics, 1999 m., T. 5.

15. *Combinatorial manifold mesh reconstruction and optimization from unorganized points with arbritrary topology.* **Huang, J ir Menq, CH.** s.l. : Computer-Aided Design, 2002 m., T. 34.

16. *A Fast Algorithm for Delaunay Based Surface Reconstruction.* **Gao, S ir Lu, H-Q.** s.l. : WSCG, 2003.

17. **Weisstein, Eric W.** Circumcircle. *A Wolfram Web Resource.* [Tinkle] [Cituota: 2008 m. Gegužės 21 d.] http://mathworld.wolfram.com/Circumcircle.html.

18. Delaunay triangulation - Wikipedia, the free encyclopedia. *Wikipedia.* [Tinkle] [Cituota: 2008 m. Gegužės 21 d.] http://en.wikipedia.org/wiki/Image:Delaunay_before_flip.png.

19. **Weisstein, Eric W.** "Delaunay Triangulation." From MathWorld--A Wolfram Web Resource. *Wolfram MathWorld.* [Tinkle] [Cituota: 2008 m. Gegužės 21 d.]

20. CGAL Open Source Project. [Tinkle] [Cituota: 2006 m. Gegužės 22 d.] http://www.cgal.org/Manual/3.3/doc_html/cgal_manual/Triangulation_3/Chapter_main.html.

21. *The Quickhull Algorithm for Convex Hulls.* **Barber, C. B, Dobkin, D. P ir Huhdanpaa, H.** 4, s.l. : ACM Transactions on Mathematical Software, 1996 m., T. 22.

22. **Garland, Michael.** *Michael Garland's web site.* [Tinkle] [Cituota: 2008 m. Balandžio 23 d.] http://graphics.cs.uiuc.edu/~garland/home.html.

23. *Multiresolution signal processing for meshes.* **Guskov, I, Schröder, P ir Sweldens, W.** New York : ACM Press/Addison-Wesley Publishing Co., 1999. 0-201-48560-5.

24. **Lin, H-W, Tai, C-L ir Wang, G-J.** A mesh reconstruction algorithm driven by an intrinsic property. *Computer-Aided Design.* s.l. : Elsevier, January 2004, T. 36, psl. 1-9.

25. **Mhatre, A ir Kumar, P.** Projective clustering and its application to surface reconstruction: extended abstract. *Symposium on Computational Geometry.* s.l. : ACM, 2006, psl. 477-478.

26. **Kuo, C-C ir Yau, H-T.** A Delaunay-based region-growing approach to surface reconstruction from unorganized points. *Computer-Aided Design.* July 2005, T. 37, psl. 825-835.

27. **O'Rourke, J.** *Computational Geometry in C, 2nd ed.* Cambridge : Cambridge University Press, 1998.

Santrauka anglų kalba

Žukas, Andrius (2008). Three dimensional objects: visualization and deformation algorithms. Master of Science graduation paper. Kaunas University of technology, department of informatics.

SUMMARY

The chosen theme of the Master of Science degree paper is “Three dimensional objects: visualization and deformation algorithms“. This subject considers surface reconstruction from point clouds and the possibilities to apply surface deformation algorithms.

During the analysis phase we found that the main problem of the algorithms of surface reconstruction from scanned point clouds is the lack of speed. So in this paper an algorithm, based on 2D Delaunay triangulation, for reverse engineering is proposed. This method divides point clouds into several parts, and then maps all the points of those point cloud parts to the plane. Then a 2D Delaunay triangulation is computed and the computed mesh is mapped back to the point cloud. We also give theoretical possibilities to apply a known algorithm for surface deformation.

During the implementation phase we found that our algorithms work as expected, but quicker than the other methods proposed earlier. We also noticed that it’s better to use 2D Delaunay triangulation for bigger point clouds and 3D Delaunay triangulation for point clouds, which contains no more than approximately 2000 points.

11. PRIEDAS NR.1

Pagrindinės įgyvendintos Matlab paketo funkcijos

Pagrindinė funkcija pėdos taškų debesies rekonstrukcijai

```
clc
clear, close all
fid = fopen('pnt5.txt', 'r');
c = fscanf(fid, '%f %f %f', [3 inf])';
fclose(fid);
c;
numrows = size(c(:,1))
r=-3;
%--------------------------------
disp('tasku perskaiciavimas');
% s=1;
% f=2;
tic
Tri1=taskai1(c,r);
Tri1;

Tes = delaunay(Tri1(:,1),Tri1(:,2));

figuros(c,Tes,r);

r=3;
s=149;
f=2;

Tri2=taskai(c,r,s,f);

Tes2 = delaunay(Tri2(:,1),Tri2(:,2));
figuros(c,Tes2,r);
axis image off
'delaunay'
tic
```

```
toc
'tasku spausdinimas'
 tic

figure
b=sortrows(c,3);
hold on
for i=1:numrows
    plot3(b(i,1),b(i,2),b(i,3),'k');
    hold on

end
%
  axis image off
```

Funkcija, skirta taškų debesies taškų paskirstymui plokštumoje

```
function B = taskai(c,r,s,f)

numrows = size(c(:,1));
tasksk=numrows/2;
B = zeros(tasksk(1,1),3);
pc(1,:)=[sorted(s,1);sorted(s,2);sorted(s,3)];
B(s,:)=[pc(1,1);pc(1,2);pc(1,3)];

for i=1:numrows/f+150
   if(i~=s)

    d=sqrt((pc(1,1)-sorted(i,1))^2+(pc(1,2)-sorted(i,2))^2+(pc(1,3)-
sorted(i,3))^2);
    fx=expm(sorted(i,3));
        d=d+fx-0.7;

    if(sorted(i,1)-pc(1,1)~=0)
         m=(sorted(i,2)-pc(1,2))/(sorted(i,1)-pc(1,1));
         % bb=sorted(i,3)-(m*sorted(i,2));
         sorted(i,1);
         p=d/sqrt(1+m^2);
         q=p*abs(m);
        if(sorted(i,1)>pc(1,1))
            xi=pc(1,1)+p;

        else
            xi=pc(1,1)-p;
        end
        if(sorted(i,2)>pc(1,2))
            yi=pc(1,2)+q;
        else
            yi=pc(1,2)-q;
        end

        B(i,:)=[xi,yi,sorted(i,3)];
    end
   end
```

```
end
```

Apatinės pėdos dalies skaičiavimas

```
function B = taskai1(c,r)

numrows = size(c(:,1));
tasksk=numrows/2;
B = zeros(tasksk(1,1),2);
% C = zeros(tasksk(1,1),3);
sorted=sortrows(c,r);
%s=159;
%sorted=c;
%pc1(1,:)=[sorted(1,1);sorted(1,2);sorted(1,3)-0.1];
pc(1,:)=[sorted(1,1);sorted(1,2);sorted(1,3)];
B(1,:)=[pc(1,1);pc(1,2)];

for i=2:numrows/2+80
%    if(i~=s)

    d=sqrt((pc(1,1)-sorted(i,1))^2+(pc(1,2)-sorted(i,2))^2+(pc(1,3)-
sorted(i,3))^2);
%     fx=expm(sorted(i,3));
%         d=d+fx-0.7;

  if(sorted(i,1)-pc(1,1)~=0)
    m=(sorted(i,2)-pc(1,2))/(sorted(i,1)-pc(1,1));
   % bb=sorted(i,3)-(m*sorted(i,2));
   sorted(i,1);
    p=d/sqrt(1+m^2);
    q=p*abs(m);
    if(sorted(i,1)>pc(1,1))
        xi=pc(1,1)+p;

    else
        xi=pc(1,1)-p;
    end
    if(sorted(i,2)>pc(1,2))
```

```
        yi=pc(1,2)+q;
    else
        yi=pc(1,2)-q;
    end

    B(i,:)=[xi,yi];
  end
end

end
```

Funkcija trianguliai atvaizduoti

```
function figuros(c,tria,r)

sorted=sortrows(c,r);
 trimesh(tria,sorted(:,1),sorted(:,2),sorted(:,3))
 hold on
%  figure
 trisurf(tria,sorted(:,1),sorted(:,2),sorted(:,3))
end
```

Funkcija, skirta sferos taškų debesies taškų perkėlimui į plokštumą

```
function B = taskai(c,r)

numrows = size(c(:,1));
tasksk=numrows/2;
B = zeros(tasksk(1,1),2);
sorted=sortrows(c,r);
%sorted=c;
pc(1,:)=[sorted(1,1);sorted(1,2);sorted(1,3)];
B(1,:)=[sorted(1,1);sorted(1,2)];
for i=2:numrows/2+25
    d=sqrt((pc(1,1)-sorted(i,1))^2+(pc(1,2)-sorted(i,2))^2+(pc(1,3)-
sorted(i,3))^2);
```

```
        d=d-0.15;
    if(sorted(i,1)-pc(1,1)~=0)
    m=(sorted(i,2)-pc(1,2))/(sorted(i,1)-pc(1,1));
    bb=sorted(i,2)-(m*sorted(i,1));

    p=d/sqrt(1+m^2);
    if(m<0&&sorted(i,1)<0)
        p=p*-1;
    end
    if(m>0&&sorted(i,1)<0)
        p=p*-1;
    end
    q=p*m;
    xi=pc(1,1)+p;
    yi=pc(1,2)+q;
    B(i,:)=[xi,yi];
    end
end

end
```

12.PRIEDAS NR.2

Siūlomi algoritmų patobulinimai

Dvimatei Delaunay trianguliacijai apskaičiuoti galėtų būti naudojamas ne Qhull metodas, o koks nors kitas. Qhull metodo blogiausias vykdymo laikas yra $O(n^2)$, todėl siūlomo algoritmo efektyvumas gali labai kristi. Aš pasirinkau Qhull metodą, nes jis turi papildomų savybių, taip pat yra lengvai įgyvendinamas.

Dar vienas patobulinimas galėtų būti įgyvendintas siekiant išvengti papildomai susidariusių trikampių. Jie atsiranda dėl to, kad Delaunay trianguliacija visų pirma suskaičiuoja iškiląjį daugiakampį, o tik po to jį dalina į atskirus trikampius. Todėl jei rekonstruojamo objekto paviršiuje yra skylių, įlinkimų ar panašių detalių, jos automatiškai patenka į iškiliojo daugiakampio vidų ir yra trianguliuojamos. Norint šito išvengti galima taikyti segmentaciją. Duotą taškų debesį suskirstyti į pakankamai mažas dalis, kurios neturėtų detalių patenkančių į iškiliojo daugiakampio vidų. Taip galima išvengti problemos, kuri pavaizduota 14 paveiksle.