VILNIAUS UNIVERSITETAS

MARTYNAS SABALIAUSKAS

NESTANDARTINĖS AVALYNĖS GAMYBOS FORMŲ PAVIRŠIŲ KOMPIUTERINIO MODELIAVIMO TECHNOLOGIJA

Daktaro disertacija

Technologijos mokslai, informatikos inžinerija (07 T)

Vilnius, 2017

Disertacija rengta 2012–2016 metais Vilniaus universitete.

Mokslinis vadovas

dr. Virginijus Marcinkevičius (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – 07 T).

Padėka

Nuoširdžiai dėkoju mokslininiam vadovui dr. Virginijui Marcinkevičiui už naudingas konsultacijas, už visapusišką pagalbą rašant disertaciją, už kantrybę ir patarimus bei nuolatinį skatinimą dirbti ir tobulėti.

Už įgytas optimizavimo metodų žinias ir už pagalbą rašant pirmuosius doktorantūros studijų straipsnius dėkoju prof. habil. dr. Jonui Mockui.

Dėkoju disertacijos recenzentams prof. dr. Juliui Žilinskui ir dr. Viktorui Medvedevui už naudingas pastabas, patarimus ir diskusijas.

Taip pat dėkoju Vilniaus universiteto Matematikos ir informatikos instituto direktoriui prof. habil. dr. Gintautui Dzemydai už visapusišką paramą studijuojant doktorantūroje.

Už pagalbą ir bendradarbiavimą norėčiau padėkoti Vilniaus universiteto Matematikos ir informatikos instituto Sistemų analizės skyriaus darbuotojams ir doktorantams, išskirtinai iš jų – dr. Aurimui Rapečkai ir Mykolui Bilinskui.

Taip pat esu dėkingas kalbos redaktorėms Audrai Ivanauskienei ir Danguolei Straižytei bei savo šeimos nariams už kantrybę ir moralinį palaikymą.

Martynas Sabaliauskas

Santrauka

Sparčiai tobulėjant 3D skenavimo technologijoms susiduriame su vis didesnės apimties taškų debesimis, todėl reikalingi efektyvūs algoritmai, skirti 3D skaitmeniniams modeliams sudaryti bei apdoroti. Šioje disertacijoje nagrinėjami 3D skaitmeninių modelių paviršių dalijimo ir gautų segmentų išklotinių sudarymo algoritmai siekiant juos panaudoti ortopedijos srityje, vienetinės avalynės gamyboje. Sėkmingas šių algoritmų pritaikymas ir įgyvendinimas pagreitintų šią gamybą, nes žmogaus atliekamas rankinis darbas būtų keičiamas programinės įrangos naudojimu.

Disertacijos tikslas – sukurti tikslų ir spartų metodą, pagal kurį automatiškai būtų sudaromi 3D skaitmeninių kurpalių modelių paviršių segmentai ir jų išklotinės, atitinkančios vienetinės avalynės gamyboje naudojamus lekalus. Tuo tikslu buvo nagrinėjami autalininai, izometriniai ir konforminiai paviršiaus išklotinių sudarymo metodai, jų rezultatai buvo lyginami su ekspertų sudarytais lekalais, kol pavyko atrasti tinkamiausią išklotinių sudarymo metodą bei sukurti ir realizuoti efektyvų kurpalių segmentavimo metodą.

Disertacijoje atlikta analitinė skaitmeninių modelių parametrizavimo ir kitų apdorojimo metodų apžvalga, įskaitant retųjų matricų apdorojimo matematinius metodus.

Išvestas analitinis matricų skaidymo singuliariosiomis reikšmėmis sprendinys panaudotas paviršių išklotinių sudarymo algoritmuose bei išklotinių deformacijos įverčiams apskaičiuoti.

Bendra disertacijos apimtis 130 puslapių, 50 numeruotų formulių, 36 paveikslai, 7 lentelės ir 7 algoritmai. Literatūros sąrašą sudaro 111 šaltinių.

Tyrimų rezultatai publikuoti 4 periodiniuose recenzuojamuose mokslo žurnaluose, 2 recenzuojamuose konferencijų leidiniuose ir 4 kitose konferencijų santraukose.

Šie rezultatai pristatyti ir aptarti 2 tarptautinėse ir 9 nacionalinėse konferencijose.

Abstract

Rapid progression of 3D scanning techniques faces the increase in higher density point clouds, therefore effective algorithms are needed to construct and process 3D Digital models. This thesis analyses the surface construction of 3D digital models and segment parameterization algorithms aimed at using them in the area of orthopedics, in the manufacturing of individual shoe models. Successful application and realization of these algorithms would facilitate the manufacturing process, while manual work would be replaced by the relevant software.

The aim of this thesis is to construct and realize an algorithm, which would automatically construct 3D last digital surface model segments and their parameterized surface, complying to the lasts used in the manufacturing of individual shoe models. Having in mind this aim authalic, isometric and conformal mapping surface parameterization methods were used, their results were compared with the shoe lasts produced by the experts, until finally the most suitable surface parameterization method was constructed and shoe last segmentation algorithm was realized.

The thesis presents an analytical overview of the methods used to process digital models, including mathematical methods of processing sparse matrix. The derived analytical matrix singular value decomposition solution was used in surface parameterization algorithms.

The general scope of the thesis — 130 pages, 50 numbered formulas, 36 figures, 7 tables and 7 algorithms. The list of references is comprised of 111 sources.

The results were published in 4 periodical scientific journals and 6 other publications. These results were presented and discussed in 2 international and 9 national conferences.

Paveikslų sąrašas

Lentelių sąrašas

Algoritmų sąrašas

Ženklai ir santrumpos

Simboliai

α	plokštuma trimatėje Euklido erdvėje
$\triangle ABC \in T_\triangle$	trikampių tinklui $T_\triangle$ priklausantis trikampis $\triangle ABC$, t. y. $Adj[A] = \{B, C, \dots\}, Adj[B] = \{A, C, \dots\}, Adj[C] = \{A, B, \dots\}$
$Adj[v]$	grafo viršūnės $v \in V$ gretimų viršūnių aibė
$\mathbf{B} = (b_{i,j})$	n-os eilės grafo $n \times n$ gretimumo matrica, $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant n$
C^k	paviršius, sudarytas iš kelių parametrinių paviršių, kurių bet kuriame tarpusavio jungimosi taške sutampa k-tos eilės išvestinės
$d(A^*, \alpha)$	atstumas nuo taško $A^*(a_1^*, a_2^*, a_3^*)$ iki plokštumos α Euklido erdvėje
$\det \mathbf{M}$	matricos $\mathbf{M}$ determinantas
$\|\mathbf{M}\|$	matricos $\mathbf{M}$ Frobenijaus norma
$\operatorname{tr} \mathbf{M}$	matricos $\mathbf{M}$ pėdsakas
$\mathbf{M}^\intercal$	transponuotoji matrica $\mathbf{M}$
$\|E\| = m$	grafo dydis m (grafo briaunų skaičius)
$G = (V, E)$	grafas G, kurį sudaro viršūnių, turinčių koordinates, aibė $V = \{v_1(x_1, y_1, z_1), v_2(x_2, y_2, z_2), \dots\}$ ir briaunų aibė E
$G^* \subset G$	grafo G pografis G^*, $V^* \subset V$, $E^* \subset E$
$\mathbf{K} = (k_{i,j})$	2×2 kovariacijos matrica, $1 \leqslant i \leqslant 2$, $1 \leqslant j \leqslant 2$
$\mathbf{L} = (l_{i,j})$	n-os eilės grafo $n \times n$ Laplaso matrica, $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant n$
$\overrightarrow{n}_{\triangle ABC}$	trikampio $\triangle ABC$ normalės vektorius
$Q = \{q_1, q_2, \dots\}$	baigtinė aibė Q, kurios elementų eiliškumas nesvarbus
$\|Q\|$	baigtinės aibės Q galia
$R = [r_1, r_2, \dots]$	sutvarkytas sąrašas R, kuriame išlaikomas elementų eiliškumas
$\|R\|$	sąrašo R elementų skaičius
$\hat{S}$	reguliarusis paviršius

s	pradinė grafo viršūnė $s \in V$ vykdant paieškos algoritmą
$u \rightsquigarrow v$	takas tarp viršūnių u ir v
$\{u, v\}$	grafo briauna, $u, v \in E$, u ir v – gretimos viršūnės
$\lvert V \rvert = n$	grafo eilė n (grafo viršūnių skaičius)
V_{int}	grafo vidinių viršūnių aibė
$T_{\triangle}$	trikampių tinklo (trianguliarusis) paviršius
$\overline{T}_{\triangle}$	trikampių tinklo paviršiaus išklotinė
$T_{\square}$	keturkampių tinklo (kvadrianguliarusis) paviršius
ω	santykinis figūrų panašumas

Santrumpos

$ABF(T_\triangle)$	trikampių kampų išlaikymu grįstas paviršiaus $T_\triangle$ išklotinės $\overline{T}_\triangle$ sudarymo algoritmas (angl. *angle based flattening*)
$ABF{+}{+}(T_\triangle)$	trikampių kampų išlaikymu ir retųjų matricų atskyrimu grįstas paviršiaus $T_\triangle$ išklotinės $\overline{T}_\triangle$ sudarymo algoritmas (angl. *angle based flattening plus plus*)
$ARAP(T_\triangle)$	trikampių kraštinių ilgius išlaikantis paviršiaus $T_\triangle$ išklotinės $\overline{T}_\triangle$ sudarymo algoritmas (angl. *as rigid as possible*)
$ASAP(T_\triangle)$	trikampių panašumą išlaikantis paviršiaus $T_\triangle$ išklotinės $\overline{T}_\triangle$ sudarymo algoritmas (angl. *as similar as possible*)
B-spline	kontroliniais taškais apibrėžiama polinominės išraiškos parametrinė kreivė
BFS(G,s)	paieška į plotį grafe G pradedant viršūne $s \in V$ (angl. *breadth-first search*)
CAD	programinė įranga, skirta daiktų ar įrenginių dizainui kurti, modeliuoti, analizuoti (angl. *computer-aided design*)
DFS(G)	paieška į gylį grafe G (angl. *depth-first search*)
Dijkstra(G,u,v)	trumpiausių takų paieškos Dijkstros algoritmas, tarp viršūnių u ir v grafe G
$LSCM(T_\triangle)$	konforminis paviršiaus $T_\triangle$ išklotinės $\overline{T}_\triangle$ sudarymo algoritmas (angl. *low stretch conformal mapping*)
NURBS	netolydieji racionalieji iš B-spline kreivių sudaryti paviršiai (angl. *non-uniform rational B-spline*)
SVD	singuliarioji matricų dekompozicija (angl. *singular value decomposition*)

Turinys

1 Įvadas

1.1 Tyrimų sritis

Fizinių objektų perkėlimas į skaitmeninius formatus taikant 3D skenavimo technologijas tapo dideliu žingsniu kuriant naujas gamybos technologijas [81, 97]. Sparčiai augant individualizuotų gaminių rinkai, reikalingi efektyvūs vienetinės gamybos (prekių gaminimo pagal individualius užsakymus) automatizavimo sprendimai [73]. Gamyboje brėžiniai ir skaitmeniniai modeliai sudaro apie 70 % techninių dokumentų, tačiau trūksta techninių sprendimų, leidžiančių efektyviai panaudoti skaitmeninius duomenis būtent individualios gamybos technologijose [37, 74]. Vienas tokių inovatyvių sprendimų – nestandartinių trimačių avalynės gamybos formų paviršių skaitmeninių modelių paviršių išklotinių sudarymo automatizavimas.

Šio darbo tyrimų sritis yra paviršių netiesinės projekcijos metodų analizė, jų taikymo ir gerinimo būdai.

1.2 Problemos aktualumas

Vienetinės ir nestandartinės avalynės gamybinių modelių (kurpalių) paviršių išklotinių sudarymas šiuo metu yra rankinis darbas. Rinkoje yra nemažai sprendimų, susijusių su automatizuota avalynės gamyba. Šie sprendimai įgyvendinami atskirose sistemose arba kaip universalių sistemų papildiniai (angl. *plugins*). Batų gamybos specialistai kaip pagalbines priemones nestandartinei avalynei modeliuoti naudoja šias programas:

1. *ShoeMaster*[1] – automatizuota avalynės modeliavimo programinė įranga, kurioje realizuoti sprendimai yra sunkiai suderinami su individualiais pėdos duomenimis. Programa skirta darbui su standartizuotais pėdos modeliais.

[1] 3D modeliavimo programinė įranga *ShoeMaster*, skirta avalynės industrijai. Internetinė prieiga: http://www.shoemaster.co.uk.

2. *Blender*[2] ir *3D Coat*[3] – laisvojo modeliavimo (angl. *freeform modeling*) programų sistemos. Jos turi sparčius paviršių išklotinių sudarymo įrankius, tačiau rezultatas yra netikslus lyginant su rankiniu būdu gautais rezultatais. *Blender* ir *3D Coat* paviršių išklotinių sudarymo įrankiais tekstūra priskiriama objektui (angl. *texture mapping*). Šios programos neturi gautų rezultatų išvesties įrankių.

3. *Terra Flat*[4] – *RhinoCeros*[5] programos papildinys. Nuskenuoti individualūs pėdos modeliai vidutiniškai yra sudaryti iš 30000–50000 trikampių, sujungtų viršūnėmis. *Terra Flat* papildinys veikia, kai didžiausias trikampių skaičius yra 2–3 kartus mažesnis. Dėl to tenka aproksimuoti pėdos geometriją. Gaunamas rezultatas skiriasi nuo tradicinio paviršiaus išklotinės sudarymo metodo, kai naudojama vaško folija, apie 30 %.

Nė viena šiandieninėje rinkoje esanti modeliavimo sistema netenkina gamybos proceso išklotinės tikslumo reikalavimo. Nestandartinių trimačių organinės prigimties formų objektų skaitmeninių modelių paviršių išklotinių sudarymo automatizavimas leistų paspartinti masinės vienetinės gamybos proceso trimačių paviršių išklotinių sudarymą.

Jeigu būtų sukurtas algoritmas, kuris automatiškai iš skaitmeninio kurpalio modelio sudarytų skaitmeninius lekalus, tada nebereikėtų atlikti šių vienetinės avalynės gamybos procesų:

1. Kurpalių paviršių segmentavimo.

2. Šių segmentų išplokštinimo naudojant vaško foliją.

3. Gamybinių brėžinių sudarymo pagal gautų vaško folijos išklotinių kontūrus.

4. Gautų gamybinių brėžinių suskaitmeninimo.

Dėl šio technologinio sprendimo modelio paruošimo gamybai laikas sutrumpėtų apie 30 %. Centralizuotos gamybos atvejais arba užsakant paslaugas nutolu-

[2] 3D modeliavimo atviroji programinė įranga *Blender*. Internetinė prieiga: www.blender.org.
[3] 3D modeliavimo programinė įranga *3D Coat*. Internetinė prieiga: http://3d-coat.com.
[4] 3D modeliavimo programinės įrangos *RhinoCeros* papildinys *Terra Flat*. Internetinė prieiga: http://www.terraflat.com.
[5] NURBS kreivių ir paviršių kompiuterinio modeliavimo programinė įranga *RhinoCeros*. Internetinė prieiga: www.rhino3d.com.

siuose gamybos centruose mažėtų gamybinių modelių (kurpalių) siuntimo išlaidos bei trumpėtų siuntimo laikas. Siuntimo išlaidos sudaro santykinai didelę viso gamybos laiko ir sąnaudų dalį, nes masinės vienetinės gamybos visi gamybiniai modeliai yra skirtingi. Sukurtas metodas leistų spręsti panašias problemas, kuriant analogiškas sistemas kitoms pramonės šakoms, kurioms yra svarbus trimačių objektų išklotinių gavimas.

Kita svarbi problema – išklotinių sudarymo algoritmų greitaveika. Kiekvieną skaitmeninį modelį apibrėžia gretimumo matrica, kurios eilė sutampa su šio skaitmeninio modelio viršūnių skaičiumi. Sudarant trianguliariojo paviršiaus išklotinę sprendžiamas optimizavimo uždavinys, kurio kiekvienos iteracijos metu apdorojama atvirkštinė gretimumo matrica, taigi iteracijos sudėtingumas tampa kvadratinis. Tačiau įvertinus, kad gretimumo matrica yra retoji matrica, taikant Choleckio dekompoziciją ar kitas matricų faktorizacijas gaunamos retosios matricos. Taigi išklotinių sudarymo algoritmo iteracijos sudėtingumas gali būti gerinamas iki tiesinio.

1.3 Tyrimų objektas

Disertacijos tyrimų objektas yra 3D skaitmeninių modelių trikampių ir keturkampių tinklo paviršių apdorojimo ir formavimo algoritmai: paviršiaus segmentavimo, paviršiaus išklotinės sudarymo, trianguliacijos, paviršiaus taškų tolygiojo paskirstymo, parametrinio paviršiaus konstravimo, grafų teorijos algoritmai. Šiems algoritmams ištirti ir realizuoti disertacijoje nagrinėjami globalaus optimizavimo, matricų dekompozicijos ir skaičiuojamosios geometrijos uždaviniai.

1.4 Tyrimų tikslas ir uždaviniai

Darbo tikslas – remiantis ortopedijos ekspertų žiniomis sukurti automatizuotą metodą, skirtą vienetinių kurpalių skaitmeninių modelių paviršių segmentavimui ir sudarytų segmentų projektavimui į plokštumą.

Šiam tikslui pasiekti sprendžiami šie uždaviniai:

1. Ištirti 3D skaitmeninių modelių paviršių formavimo ir apdorojimo algorit-

mus.

2. Ištirti parametrinių paviršių sudarymo metodus, kuriuos būtų galima pritaikyti formuojant parametrinius paviršius iš taškų debesų (skaitmeninių modelių), gautų iš nuskenuotų kurpalių.

3. Ištirti, kurie trikampių tinklo paviršių išklotinių sudarymo metodai labiausiai minimizuoja paviršių deformacijas.

4. Ištirti retųjų matricų apdorojimo metodus, kuriais greitinamas paviršių išklotinių sudarymas.

5. Įvertinti, kurio iš standartinių paviršių parametrizavimo metodų (autalinių, konforninių ar izometrinių) rezultatas labiausiai atitinka vienetinės avalynės gamyboje rankiniu būdu sudaromus lekalus.

6. Sukurti automatinį kurpalių skaitmeninių modelių paviršių segmentavimo metodą, tinkantį nuskenuotų kurpalių paviršių išklotinėms sudaryti.

7. Palyginti rankinio ir sukurto automatinio kurpalių skaitmeninių modelių paviršių segmentavimo metodų rezultatus.

8. Sukurti programinę įrangą, skirtą nestandartinių avalynės formų paviršių segmentavimui ir šių segmentų paviršių išklotinių sudarymui.

1.5 Tyrimų metodai

Disertacijoje suformuluoti uždaviniai sprendžiami analitiniais ir eksperimentiniais metodais. Analizuojant mokslinius ir eksperimentinius rezultatus skaitmeninių modelių paviršių segmentavimo ir išplokštinimo srityse naudojami informacijos paieškos, sisteminimo, analizės ir apibendrinimo metodai. Teoriniai tyrimo metodai taikomi tiriant algoritmų sudėtingumą ir spartinimą. Apibendrinimo metodu įvertinami statistikos metodais gautos išklotinės palyginimo su vienetinės avalynės gamyboje naudojamais lekalais rezultatai.

1.6 Mokslinis naujumas

Disertacijos rezultatai:

1. Darbe atlikti eksperimentiniai tyrimai, skirti įvertinti kurpalių lekalų, gautų skirtingais paviršių išklotinių sudarymo algoritmais, atitikimui su lekalais, gautais vienetinės avalynės gamybos metu.

2. Išvesti analitiniai 2×2 matricų skaidymo singuliariosiomis reikšmėmis sprendiniai panaudoti lokalioje ARAP algoritmo optimizavimo fazėje, o Choleckio dekompozicija pritaikyta atvirkštinei retajai Laplaso matricai apskaičiuoti globalioje ARAP algoritmo optimizavimo fazėje. Visa tai leido iš esmės paspartinti izometrinį paviršių išklotinių sudarymo ARAP algoritmą.

3. Sukurtas naujas automatinis kurpalių skaitmeninių modelių paviršių segmentavimo algoritmas, pagal kurį sudarytų segmentų išklotinės apytiksliai 98 % santykiniu panašumu sutampa su vienetinės avalynės gamyboje naudojamais lekalais.

4. Kurpaliams modeliuoti pasiūlytas naujas kvadrianguliariųjų konstrukcijų aproksimavimo Bezjė paviršiais, atitinkančiais parametrinį C^2 paviršių, algoritmas.

1.7 Praktinė darbo reikšmė

Tikslus ir spartus skaitmeninių modelių paviršių ir jų fragmentų išplokštinimas teiktų didelę naudą daugeliui pramonės sričių: avalynės, tekstilės, automobilių, baldų, medicinos gaminių gamybos pramonei. Spartus virtualiųjų modelių paviršių plokštinimas paspartintų vienetinių ir unikalių produktų gamybą bei suteiktų daugiau technologinių galimybių ir gamybos proceso efektyvumo ypač toms pramonės sritims, kurios dirba su 3D skenavimo technologijomis ir gamina masinius individualius (angl. *mass customization*) bei vienetinius produktus.

Šioje disertacijoje pasiūlytas kurpalių skaitmeninių modelių paviršių segmentavimo algoritmas kartu su izometriniu paviršių išklotinių sudarymo algoritmu gali būti naudojami vienetinės avalynės gamyboje sudarant paviršių 2D brėžinius (lekalus).

Disertacijoje atliktas darbas, kuriame sukurti praktiškai pritaikomi algoritmai, 6-ojoje LMA jaunųjų mokslininkų konferencijoje „Fizinių ir technologijos mokslų tarpdalykiniai tyrimai“ įvertintas aukščiausiu asociacijos *INFOBALT* apdovanojimu, skirtu jauniesiems mokslininkams.

1.8 Ginamieji teiginiai

1. Sukurtu automatiniu kurpalių skaitmeninių modelių paviršių segmentavimo algoritmu, naudojant izometrinį paviršių išklotinių sudarymo ARAP algoritmą, apskaičiuojamos išklotinės labiau atitinka vienetinės avalynės gamyboje naudojamus lekalus nei išklotinės, apskaičiuojamos rankiniu būdu segmentuojant kurpalių paviršių ir naudojant konforminius ABF++ ir LSCM paviršių išklotinių sudarymo algoritmus.

2. Naudojant izometrinį kurpalių skaitmeninių modelių segmentų išklotinių sudarymo ARAP algoritmą bendra visų trikampių ploto ir panašumo deformacija yra minimizuojama, skirtingai nei naudojant konforminius ABF++ ir LSCM paviršių išklotinių sudarymo algoritmus.

3. Sukurtas ir realizuotas Bezjė paviršių aproksimavimo algoritmas kvadrianguliariesiems įvesties paviršiams pasižymi šiomis savybėmis:

   a) ypatingųjų taškų srityse apskaičiuojamos viršūnės išsidėsto tolygiai,

   b) galima laisvai pasirinkti parametrinio paviršiaus detalumo parametrą $n = 1, 2, 3, \ldots$

4. Pasiūlyti analitiniai 2×2 realiosios matricos $\mathbf{A}$ singuliariosios dekompozicijos $\mathbf{U\Sigma V}$ sprendiniai tinka apskaičiuoti trikampių tinklo paviršiaus deformacijos įverčius lokalioje ARAP paviršių išklotinių sudarymo algoritmo optimizavimo fazėje.

1.9 Darbo rezultatų aprobavimas

Tyrimų rezultatai buvo paskelbti 4 periodiniuose recenzuojamuose ir 6 kituose leidiniuose. Straipsniai periodiniuose recenzuojamuose leidiniuose:

1. **Sabaliauskas M.**, Marcinkevičius V. An investigation of ABF++, LSCM, and ARAP methods for parameterization of shoetrees. *Science – Future of Lithuania / Mokslas – Lietuvos Ateitis*, Vilnius, Technika, ISSN 2029-2341, 2015, 7(3): 296–299.

2. **Sabaliauskas M.** An algorithm for approximation of Bezier surfaces for special case of quadrangular grid. *Computional Science and Techniques*, Klaipėda, Klaipėda University, ISSN 2029-9966, 2015, 3(2): 454–563.

3. **Sabaliauskas M.**, Mockus J. On the Nash equilibrium in the inspector problem. *Lietuvos matematikos rinkinys*, ISSN 0132-2818, 2014, 55: 79–84.

4. Mockus J., **Sabaliauskas M.** On the exact polynomial time algorithm for a special class of bimatrix game. *Lietuvos matematikos rinkinys*, ISSN 0132-2818, 2013, 54: 85–90.

Straipsniai recenzuojamuose konferencijų leidiniuose:

1. **Sabaliauskas M.**, Marcinkevičius V. Segmentation model for flattening of individual 3D lasts. *EMC 2016: 6th International symposium „Engineering Management and Competetiveness“: proceedings*, Kotor, Montenegro, ISBN: 9788676722846, 2016, 325–331.

2. **Sabaliauskas M.** A robust SVD algorithm for ARAP method. *EMC 2015: 5th International symposium „Engineering Management and Competetiveness“: proceedings*, Zrejanin, Serbia, University of Novi Sad, ISBN 9788-676722563, 2015, 333–336.

Santraukos konferencijų leidiniuose:

1. **Sabaliauskas M.**, Marcinkevičius V. An investigation of surface deformation using singular value decomposition. *Data analysis methods for software systems: 8th International Workshop*: [abstracts book], Druskininkai, Lithuania, ISBN 9789986680611, 2016, p. 52.

2. **Sabaliauskas M.** Skaitmeninių organinės prigimties kurpalių modelių segmentavimo ir projektavimo į plokštumą tyrimas. *Fizinių ir technologijos mokslų tarpdalykiniai tyrimai: šeštoji jaunųjų mokslininkų konferencija:*

pranešimų santraukos, Vilnius, 2016 m. vasario 10 d. Vilnius, Lietuvos mokslų akademijos leidykla, 2016, 57–60.

3. **Sabaliauskas M.**, Marcinkevičius V. An Application of ARAP Method for Parameterization of Shoetrees. *Data analysis methods for software systems: 7th International Workshop*: [abstracts book], Druskininkai, Lithuania, ISBN 9789986680581, 2015, 45–46.

4. **Sabaliauskas M.**, Marcinkevičius V. Comparison of mods of shoetrees obtained by theoretical and experimental methods. *Data analysis methods for software systems: 6th International Workshop*: [abstracts book], Druskininkai, Lithuania, 2014 [Vilnius], ISBN 9789986680505, 2014, 44.

Skaityti pranešimai 2 tarptautinėse konferencijose:

1. 6-oji tarptautinė konferencija „Engineering Management and Competitiveness“. Kotoras, Juodkalnija. 2016 m. birželio 17–18 d.

2. 5-oji tarptautinė konferencija „Engineering Management and Competitiveness“. Zrenianinas, Serbija. 2015 m. birželio 19–20 d.

Skaityti pranešimai 9 respublikinėse konferencijose:

1. 8-oji mokslinė konferencija „Duomenų analizės metodai programų sistemoms“. Druskininkai, Lietuva. 2016 m. gruodžio 1–3 d.

2. 6-oji LMA jaunųjų mokslininkų konferencija „Fizinių ir technologijos mokslų tarpdalykiniai tyrimai“. Vilnius, Lietuva. 2016 m. vasario 10 d.

3. 7-oji mokslinė konferencija „Duomenų analizės metodai programų sistemoms“. Druskininkai, Lietuva. 2015 m. gruodžio 3–5 d.

4. 17-oji mokslinė kompiuterininkų konferencija „Kompiuterininkų dienos – 2015“. Panevėžys, Lietuva. 2015 m. rugsėjo 17–19 d.

5. 6-oji mokslinė konferencija „Duomenų analizės metodai programų sistemoms“. Druskininkai, Lietuva. 2014 m. gruodžio 4–6 d.

6. 6-oji Lietuvos jaunųjų mokslininkų konferencija „Operacijų tyrimas ir taikymas“. Vilnius, Lietuva. 2014 m. rugsėjo 22 d.

7. 55-oji Lietuvos matematikų draugijos konferencija. Vilnius, Lietuva. 2014 m. birželio 26–27 d.

8. 5-asis mokslinis seminaras „Duomenų analizės metodai programų sistemoms“. Druskininkai, Lietuva. 2013 m. gruodžio 5–7 d.

9. 54-oji Lietuvos matematikų draugijos konferencija. Vilnius, Lietuva. 2013 m. birželio 19–20 d.

Darbų pripažinimas:

1. Geriausias 6-osios LMA jaunųjų mokslininkų konferencijos „Fizinių ir technologijos mokslų tarpdalykiniai tyrimai“ technologijos mokslų sekcijos pranešimas. Vilnius, 2016 m. vasario 10 d.

2. INFOBALT asociacijos stipendija už mokslo tiriamąjį darbą „Skaitmeninių organinės prigimties kurpalių modelių segmentavimo ir projektavimo į plokštumą tyrimas“. Vilnius, 2016 m. vasario 17 d.

1.10 Disertacijos struktūra

Disertaciją sudaro 5 skyriai, literatūros sąrašas ir priedai. Disertacijos skyriai: Įvadas, Skaitmeninių modelių taikymo ir apdorojimo metodų apžvalga, Nauji 3D skaitmeninių modelių apdorojimo metodai, Išklotinių sudarymo eksperimentiniai tyrimai, Bendrosios išvados. Papildomai disertacijoje pateiktas paveikslų, lentelių, algoritmų, simbolių ir ženklų ir santrumpų sąrašas. Visa disertacijos apimtis 130 puslapių, pateikti 36 paveikslai, 7 lentelės ir 7 algoritmai. Disertacijoje remtasi 111 literatūros šaltinių.

2 Skaitmeninių modelių taikymo ir apdorojimo metodų apžvalga

3D skaitmeninių modelių kūrimo programos, pavyzdžiui, *3DS Max*[6], *Blender* ar *3D Coat*, naudoja sudėtingus algoritmus, kuriais apdorojami šie 3D modeliai. Siekiant atlikti eksperimentinius tyrimus ir išvengti geometrinių, topologinių ir kitų reikšmingų klaidų statistinius duomenis atitinkantys skaitmeniniai kurpalių modeliai buvo apdoroti 3D modeliavimo programomis: pašalintos paviršiaus skylės, dubliuoti ar labai artimi taškai, susiklojantys paviršiaus trikampiai. Taigi šiame skyriuje apžvelgsime skaitmeninių modelių apdorojimo algoritmus: trikampių tinklo sudarymo (trianguliacijos), paviršiaus triukšmo šalinimo, viršūnių skaičiaus keitimo išlaikant modelio formą, deformacijos ir kitus. Daugiau dėmesio skirsime paviršiaus išklotinių sudarymo algoritmams, jų greitaveikos problemoms spręsti.

2.1 3D skaitmeniniai modeliai ir jų apdorojimo algoritmai

Dar prieš atsirandant kompiuteriams, XVIII amžiuje, pradėta vystyti nauja matematikos sritis – grafų teorija, kurios pradininku laikomas L. Oileris. Šis matematikas pirmasis suformulavo ir išsprendė *Septynių Karaliaučiaus tiltų* uždavinį [90] vartodamas *grafo* sąvoką. Grafas šioje teorijoje buvo apibrėžtas aibe objektų (viršūnių) ir aibe sąryšių (briaunų) tarp šių viršūnių.

Tipiniai grafų teorijos uždaviniai reikalauja algoritmo, pagal kurį atliekami tam tikri skaičiavimai, kol gaunamas rezultatas, todėl grafų teorija ypač susidomėta XX amžiuje atsiradus pirmoms elektroninėms skaičiavimo mašinoms.

[6] 3D modeliavimo atviroji programinė įranga *3DS Max*. Internetinė prieiga: http://www.autodesk.com/products/3ds-max/overview.

Pirmieji grafų paieškos algoritmai buvo sukurti siekiant atrasti visas iš fiksuotos viršūnės pasiekiamas viršūnes keliaujant briaunomis [55,98], rasti trumpiausius takus tarp viršūnių [31], sukonstruoti minimalius jungiančiuosius medžius [45, 52] ir t. t. Vėliau, pradėjus sparčiai vystytis kompiuterinei grafikai, grafų teorija pradėta taikyti kuriant 3D skaitmeninių modelių apdorojimo algoritmus.

1 apibrėžimas. Skaitmeniniu modeliu vadinamas daugiakampių tinklo paviršius, kurį inicializuojant dažnai naudojami su šiuo modeliu susieti kompiuterinės grafikos elementai: atspindžio efektai, paviršiaus tekstūros, animacijos elementai, deformacijos algoritmai ir t. t.

Sparčiai tobulėjant 3D objektų skaitmeninimo technologijoms, pavyzdžiui, kompiuterinei tomografijai, 3D lazeriniam skenavimui, taip pat skenavimui pagal magnetinį rezonansą ar ultragarsą, didėja duomenų perdavimo srautai bei duomenų kiekis, todėl reikalingi efektyvūs algoritmai, metodai ir programinė įranga šiems duomenims apdoroti [16]. Pagal vystymosi raidą ir algoritmų sudėtingumą 3D skaitmeninių modelių apdorojimo algoritmai skirstomi į tris dalis:

1. *Pirminiai algoritmai* (angl. *low-level algorithms*) apima bazinius 3D skaitmeninių modelių sudarymo ir apdorojimo metodus – paviršiaus sudarymo (trianguliacijos), CAD (angl. *computer-aided design*) modeliavimo, triukšmo, geometrinių ir topologinių klaidų šalinimo, paviršiaus kokybės analizės.

2. *Vidutinio lygio algoritmai* (angl. *mid-level algorithms*) skirti 3D skaitmeninių modelių kokybei pagerinti – sukonstruoti parametriniam paviršiui, tolygiai paskirstyti bei pašalinti nereikalingoms skaitmeninio modelio viršūnėms išlaikant paviršiaus formą, sudaryti paviršiaus išklotinėms, skirtoms tekstūroms išsaugoti 2D formatu bei atkurti 3D skaitmeninių modelių inicializavimo metu.

3. *Aukšto lygio algoritmai* (angl. *high-level algorithms*) suteikia stacionariems 3D skaitmeniniams modeliams dinamiškumo. Tai skeleto sudarymo ir paviršiaus klasterizavimo metodai, kurie siejami su paviršiaus deformacijos algoritmais. Šie algoritmai daugiausia naudojami kuriant šiuolaikinius animacinius filmus ir kompiuterinius žaidimus.

2.1.1 Skaitmeninių modelių pirminiai apdorojimo algoritmai

Vieną pirmųjų skaitmeninių modelių sudarymo algoritmų iš nestruktūruoto taškų debesies 1984 m. pasiūlė J. D. Boissonnatas [13]. Šis trikampių tinklo formavimo algoritmas buvo pavadintas Delonė trianguliacija (angl. *Delaunay sculpting*). Algoritmo esmė – paviršiaus nesudarančių tetraedrų šalinimas iš formuojamo trikampių tinklo. Vėliau, 1999 m. F. Bernardinis ir kiti pasiūlė greitesnį ir efektyvesnį metodą sujungiant artimiausias viršūnes tuščių sferų principu (angl. *ball-pivoting*) [9].

Paviršių susiklojimo šalinimo algoritmą suglaudinant paviršių 1994 m. pasiūlė G. Turkas ir kiti [99]. 1996 m. B. Curlesso ir M. Levoy'aus pasiūlytas tūrio išlaikymo principu veikiantis paviršių susiklojimo šalinimo metodas [24].

Paviršiaus skylių šalinimo algoritmą, randantį mažiausio svorio trianguliaciją planariesiems daugiakampiams, pirmasis 1980 m. pasiūlė G. Kliencsekas [51]. Vėliau, 2002–2005 m., autorių P. Liepos, J. Podolako, J. Davio, M. Pauly ir kitų pasiūlyti skylių užpildymo ir paviršiaus išlyginimo metodai [26, 59, 75, 78].

XX a. paskutiniame dešimtmetyje CAD modeliavimo programose įsigalėjus NURBS paviršiams, sudarytiems iš racionaliųjų Bezjė paviršių sąjungos, daug mokslinio dėmesio buvo skiriama NURBS paviršių apdorojimui. G. Barequet ir M. Shariras pasiūlė geometrinį maišos metodą NURBS paviršių dalims tarpusavyje jungti [7]. Barequet ir S. Kumaras sukūrė CAD modeliavimo algoritmą, skirtą artimoms briaunoms sujungti [6]. P. Borodinas ir R. Kleinas pasiūlė viršūnių ir briaunų jungimo operatorių, skirtą paviršiaus tarpams pašalinti [15]. S. Bischoffas ir L. Kobbeltas pasiūlė tūrį išlaikantį NURBS paviršių apjungimo metodą [10].

Kita opi problema – paviršiaus susiklojimas (angl. *triangle soups*). Paviršiumi susikloję skaitmeniniai modeliai dažniausiai suformuojami CAD modeliavimo įrankiais neturint pakankamai informacijos apie skaitmeninio modelio struktūrą. 1997 m. T. Muralis ir kiti [67] susiejo trianguliarųjį, save kertantį paviršių su binariuoju medžiu, pagal kurio lapus nustatomi sienų tarpusavio susikirtimai. F. Nooruddinas, C. Shenas, S. Bischoffas, L. Kobbeltas, A. Gressas, R. Kleinas [11, 39, 71, 87] 2003–2005 metais naudodamiesi šiuo binariuoju medžiu pasiūlė metodus, skirtus save kertančiam paviršiui ištaisyti.

Formuojant paviršių iš taškų debesies dažnai pasitaiko įvairių geometrinių ir topologinių klaidų. Šioms klaidoms šalinti 2001–2004 m. Z. Woodas ir kiti pasiūlė skaitmeninių modelių rankenų šalinimo algoritmus, skirtus aukštesnės eilės toro laipsniui redukuoti [41,105], jei skaitmeniniam modeliui topologiškas toras turi per daug rankenų. 2001 m. A. Gueziecas ir kiti pasiūlė metodą skaitmeninių modelių, topologiškų vienmatei daugdarai, supaprastinimui, redukuojant tarpusavyje artimų viršūnių skaičių [40]. Autoriai F. Nooruddinas, R. M. Haralickas, S. Bischoffas ir kiti pasiūlė tūrį išlaikančius paviršiaus skylių užpildymo algoritmus [11, 43, 71].

Pramonėje labai svarbi gaminio paviršiaus forma: aptakesnės formos automobiliai ar lėktuvai ilgaamžiškesni ir turi mažesnį oro pasipriešinimą (čia figūruoja C^2 ar aukštesnės eilės paviršiai C^k, kur $k \in \mathbb{N}$ parodo, kiek kartų bet kuris paviršiaus taškas yra diferencijuojamas). Tuo tikslu buvo kuriami algoritmai, skirti paviršiaus kokybei įvertinti. 1994 m. H. Burchardas ir kiti [18] pasiūlė metodą paviršiaus lygumui nustatyti formule

$$\int_{\Omega} (\kappa_1^2 + \kappa_2^2) dA,$$

čia κ_1, κ_2 – skirtingų krypčių kreivumai srityje $\Omega \in \tilde{S}$. 1997 m. C. Yifanas ir kiti [106] pasiūlė paviršiaus atspindžio linijas (angl. *reflection lines*), vizualiai parodančias paviršiaus kreivumą (3.18 paveikslas). H. Hagenas, B. Carbalas, J. R. Shewchukas ir kiti [19, 42, 89] pasiūlė metodus, kuriais vizualiai matomas kampuotų skaitmeninių modelių išlygintas paviršius.

Skenuojant trimačius objektus dažnai užfiksuojami paviršiui nepriklausantys taškai ir taip atsiranda paviršiaus triukšmas, todėl buvo kuriami triukšmo šalinimo ir paviršiaus lyginimo (angl. *mesh smoothing*) algoritmai. 1994 m. N. Sapidis pasiūlė metodą [84] paviršiaus triukšmo lygiui įvertinti. 1997 m. F. Chungas įvedė spektrinės grafų teorijos sąvoką, apibendrinančią diskrečią Furjė transformaciją ir reiškiančią paviršiaus projekciją į tiesinę erdvę pagal lokaliuosius tikrinius vektorius [23]. G. Taubinas 1995–2000 m. pasiūlė spektrinę dekompoziciją, ekvivalenčią diskrečiajai kosinusinei transformacijai [94, 95], pagal kurią sukurtas paviršių išlyginimo metodas [96]. Darbuose [50, 93] spektrinė analizė pritaikyta skaitmeniniams modeliams suglaudinti redukuojant viršūnių skaičių. Darbuose [33, 102] pasiūlyti multirezoliuciniai metodai ir algebrinės transformacijos taikant spektrinę analizę skaitmeniniams paviršiams išlyginti. Autorių M. Reutero, B. Levio, M.

Desbruno, Y. Ohtakės ir kitų [30, 57, 72, 80] pasiūlyti paviršių išlyginimo algoritmai paremti Laplaso ir Beltramio operatoriaus pritaikymu paviršiaus kreivumui sumažinti.

2.1.2 Skaitmeninių modelių vidutinio lygio apdorojimo algoritmai

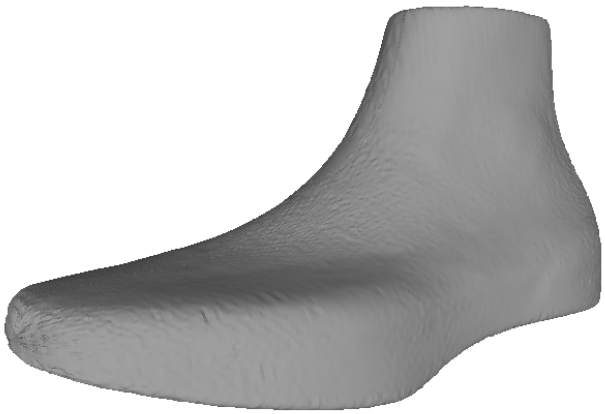
2005 m. P. Alliezas pirmasis suformulavo skaitmeninio modelio viršūnių pertvarkymo uždavinį išlaikant paviršiaus formą ir susiejo jį su paviršiaus kokybės nustatymo algoritmais, taip pat pasiūlė sprendimą trikampių tinklo paviršiams [4]. Duotą 3D skaitmeninį modelį reikia aproksimuoti kitu skaitmeniniu modeliu taip, kad paviršiaus kokybė išliktų kiek galima artima pradiniam modeliui. Angliškai toks uždavinys vadinamas *remeshing*. Vienas pirmųjų tokio pobūdžio algoritmų pasiūlytas dar 1997 m. autoriaus J. R. Shewchuko [88]. Čia įvesta izotropijos sąvoka, kuri parodo skaitmeninio modelio sienos ribojamo ploto ir kraštinių ilgių sumos priklausomybę. Aukštos izotropijos skaitmeninio modelio viršūnės išdėstytos taip, kad kiekviena siena ribotų kuo didesnį plotą. Šios izotropijos didinimo principu buvo pasiūlyti tolygiojo viršūnių paskirstymo paviršiuje algoritmai [88, 89]. 2005 m. pasiūlyti godieji viršūnių paskirstymo metodai Delonė trianguliacijos viršūnių jungimo principu [14, 28]. 1999–2004 m. pasiūlyti variaciniai viršūnių paskirstymo metodai [5, 21, 35] optimizuojant $T_{\triangle}$ paviršiaus viršūnių išdėstymą pagal trikampių kampus, plotus ir kraštinių ilgius. Kiti viršūnių paskirstymo metodai [3, 34, 64, 76, 101] paremti Voronojaus ląstelių siejimu su skaitmeninio modelio sienomis bei šių ląstelių centrų tolygiu paskirstymu.

Šiuolaikinė kompiuterinė grafika nesuvokiama be tekstūrų. Joms išsaugoti naudojami 2D paveikslėlių tipo formatai, kurie 3D skaitmeninių modelių inicializacijos metu užtempiami ant šių 3D modelių paviršių. Tam tikslui pasiekti reikalingi 3D skaitmeninių modelių paviršių išklotinių sudarymo algoritmai. Pagal išklotinės kontūro viršūnių išdėstymą metodai skirstomi į 2 dalis:

1. Fiksuotųjų kraštinių viršūnių išklotinės sudarymas (angl. *barycentric mapping*). Paviršiaus išklotinė dedama į taisyklingą daugiakampį, o kraštinės viršūnės išdėstomos ant šio daugiakampio kraštinių [36, 38].

2. Laisvųjų kraštinių viršūnių išklotinės sudarymas (angl. *free boundary mapping*). Pagal šį metodą $T_{\triangle}$ išklotinė sudaroma trimis būdais: išlaikant trikampių kampus [58, 69, 85] (angl. *conformal mapping*), trikampių kraštinių ilgius [44,62,83] (angl. *isometric mapping*) ir trikampių plotus (angl. *authalic mapping*). Šiuos metodus disertacijoje aptarsime detaliau 2.2 skyriuje.

Siekiant aproksimuoti skaitmeninius modelius matematiniais C^2 paviršiais ir sumažinti šių modelių apimtį, sudaromi parametriniai paviršiai. Lietuvoje 2013 m. *A. Davidsonas* apgynė disertaciją „Parametrinio paviršiaus modelio sudarymas iš trimačio taškų debesies“ [25]. Disertacijoje pasiūlytas netolydžiųjų B-spline kreivių jungimo metodas pavadintas NUBS (angl. *Non-Uniform B-Spline Surface*), kurio pagrindinė idėja – sukonstruoti parametrinį paviršių trimačiam taškų debesiui, jei jame galima sukonstruoti atkarpą, iš kurios išvesti ortogonalieji spinduliai paviršių kerta ne daugiau kaip vieną kartą. 2.1 paveiksle pateiktas NUBS metodu suformuotas parametrinis paviršius. Taip pat iš Lietuvos mokslininkų verta paminėti *K. Karčiauską*, pasiūliusį kitų metodų, skirtų C^2 paviršiams sudaryti [47–49].



2.1 paveikslas. NUBS metodu sudarytas parametrinis kurpalio paviršius iš trimačio taškų debesies

2.1.3 Skaitmeninių modelių aukšto lygio apdorojimo algoritmai

Siekiant sukonstruoti parametrinį skaitmeninio modelio paviršių visų pirma reikalingas topologinis įvertis, pagal kurį sudaroma atskaitos sistema. Ši proble-

ma sprendžiama sudarant skaitmeninio modelio skeletą. Pagrindinė skeleto sudarymo idėja – palaipsniui mažinti įvesties modelio tūrį kiek įmanoma išlaikant paviršiaus formą, kol galiausiai gaunamas skeletas [2, 54, 63]. Ši problema dar neturi visuotinai priimto sprendimo, nes paviršiaus formų esama įvairių, todėl nėra sukurto universalaus skeleto sudarymo algoritmo.

Sudarant skaitmeninių modelių paviršių išklotines svarbu kiek galima sumažinti paviršiaus deformaciją. Tuo tikslu 3D modelių paviršius skaidomas į segmentus pagal paviršiaus kreivumą ar gradiento kryptį, viršūnių tankumą, spalvų išdėstymą ir kitus kriterijus [1, 8, 22]. Lietuvių mokslininkai A. Lipnickas ir V. Raudonis pagal paviršiaus kreivumą pasiūlė paviršiaus sričių klasterizavimo metodą segmentams sudaryti [61]. Eksperimentiniai segmentavimo tyrimai buvo atlikti naudojant kubo ir žmogaus pėdos skaitmeninius modelius.

Atskirų segmentų skaitmeninio modelio skeleto sudarymas įgalina sukonstruoti ir susieti atskirus parametrinius paviršius, o judinant šių parametrinių paviršių kontrolinius taškus sudaroma kompiuterinė animacija [109]. Tokio pobūdžio algoritmai vadinami paviršiaus deformacijos algoritmais. Iš šių metodų verta paminėti gradientą išlaikančią deformaciją taikant mažiausių kvadratų metodą [107, 108], Laplaso koordinates išlaikančius metodus [60, 68, 92, 111] ir prizminį deformacijos metodą [17].

Kitame poskyryje apžvelgsime paviršių išklotinių sudarymo principus ir metodus.

2.2 Paviršių išklotinės

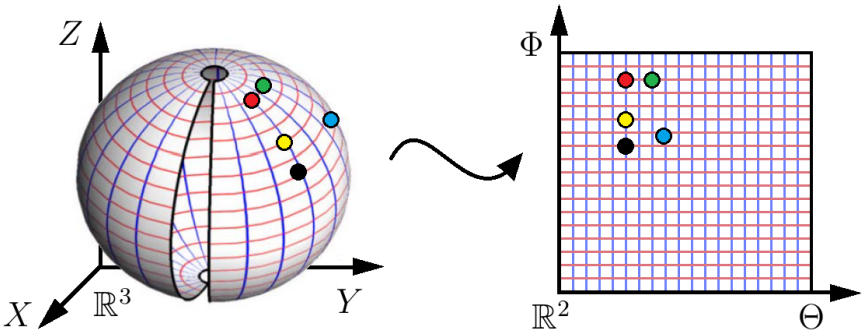
Paviršių išklotinių sudarymas nuo antikos laikų siejamas su *žemėlapių sudarymu*. Iš čia ir kilęs angliškas terminas *mapping*. Vėliau, kai buvo išsiaiškinta, jog Žemė yra rutulio formos, iškilo poreikis jos trimatį paviršių kuo mažiau deformuojant atvaizduoti plokštumoje.

Vienas iš būdų matematiškai atvaizduoti sferą plokštumoje – sferinių koordi-

načių

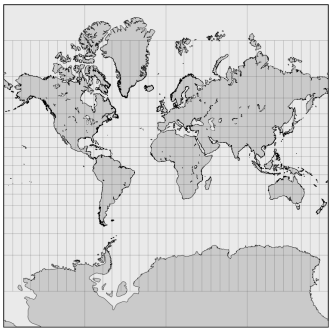
$$\begin{cases} x(\theta,\phi) = r\cos(\theta)\cos(\phi), \\ y(\theta,\phi) = r\sin(\theta)\cos(\phi), \\ z(\theta,\phi) = r\sin(\phi) \end{cases} \tag{2.1}$$

parametrų $\theta \in [0; 2\pi), \phi \in [-\frac{\pi}{2}; \frac{\pi}{2})$ susiejimas su stačiakampe Dekarto koordinačių sistema. Kitaip tariant, sferą galima atvaizduoti polinėje koordinačių sistemoje (2.2 paveikslas). Šią cilindrinę žemėlapio projekciją pirmasis pasiūlė fla-



2.2 paveikslas. Sferos atvaizdavimas plokštumoje

mandų kilmės ortografas *Gerardus Mercatorius* 1569 m. Kad ir koks patogus šis metodas matematine prasme, tačiau plokštumoje atvaizduotas žemėlapis tampa labiausiai deformuotas arti Žemės polių (2.3a paveikslas). Siekiant išspręsti šią



(a) Mercatoriaus Žemės paviršiaus projekcija

(b) Stereografinis žemėlapis

(c) Lamberto Žemės paviršiaus projekcija

2.3 paveikslas. Žemės paviršiaus išklotinių sudarymas

problemą buvo pasiūlyta stereografinė žemėlapio projekcija. Pagal šią projekciją žemės pusrutulių išklotinės sudaromos išlaikant žemynų kontūrų panašumą (2.3b paveikslas). Ši projekcija dar vadinama konforminiu žemėlapio sudarymu (angl. *conformal mapping*) [32]. 1789 m. vokiečių matematikas *J. H. Lambertas* pasiūlė matematinį metodą sferos išklotinei sudaryti lokaliai išlaikant paviršiaus ploto

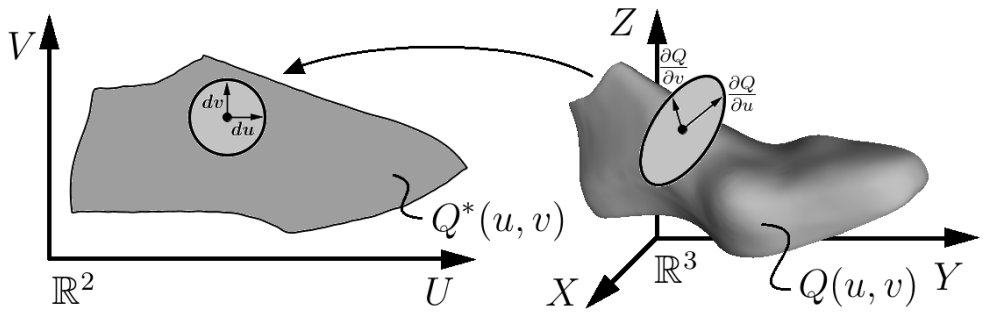
paskirstymą. Šis metodas dar vadinamas Lamberto azimutine lygaus ploto projekcija [104] (2.3c paveikslas) arba kitaip – autaliniu žemėlapio sudarymu (angl. *authalic mapping*) [56]. Šiuo metu yra šimtai matematinių metodų, skirtų Žemės paviršiaus išklotinei sudaryti, tačiau dauguma jų – apžvelgtų metodų kompozicijos.

2.2.1 Paviršių parametrizavimo metodai

Šiame skyriuje apibendrinsime išklotinės sudarymą, t. y. nagrinėsime bet kokio paviršiaus parametrizavimo metodus remdamiesi diferencialine geometrija.

2 apibrėžimas. Paviršiaus parametrizavimu vadinamas skaitmeninio modelio ar kito geometrinio objekto apdorojimo procesas, kurio metu paviršiui priskiriama parametrinė jo išraiška, pavyzdžiui, (u, v) koordinatės.

Pagrindinė užduotis – kiek įmanoma sumažinti deformaciją sudarant paviršiaus išklotinę (2.4 paveikslas).



2.4 paveikslas. Parametrinio paviršiaus išklotinės sudarymo pavyzdys

3 apibrėžimas. Reguliariuoju paviršiumi vadinamas parametrinis paviršius

$$\hat{S}(u, v) = (x(u, v), y(u, v), z(u, v)),$$

jei tenkinamos šios sąlygos:

1. Funkcijos $x(u, v), y(u, v), z(u, v)$ turi pirmos eilės išvestines.
2. Paviršiaus $\hat{S}(u, v)$ tangentiniai vektoriai $\hat{S}_u = \frac{\partial \hat{S}}{\partial u}$ ir $\hat{S}_v = \frac{\partial \hat{S}}{\partial v}$ yra tiesiškai nepriklausomi.

Toliau nagrinėsime reguliarųjį paviršių

$$Q(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}. \tag{2.2}$$

Keičiant parametrinio paviršiaus $Q(u,v)$ kintamuosius, transformaciją iš plokštumos į trimatę erdvę apibrėžia 3×2 dalinių išvestinių Jakobio matrica

$$\mathbf{J} = \begin{pmatrix} \frac{\partial x(u,v)}{\partial u} & \frac{\partial x(u,v)}{\partial v} \\ \frac{\partial y(u,v)}{\partial u} & \frac{\partial y(u,v)}{\partial v} \\ \frac{\partial z(u,v)}{\partial u} & \frac{\partial z(u,v)}{\partial v} \end{pmatrix}. \tag{2.3}$$

Pagal šią Jakobio matricą lokaliai paviršių $Q(u,v)$ charakterizuoja pirmoji fundamentalioji forma

$$\mathbf{I} = \mathbf{J}^\intercal \mathbf{J} = \begin{pmatrix} \frac{\partial Q^\intercal}{\partial u}\frac{\partial Q}{\partial u} & \frac{\partial Q^\intercal}{\partial u}\frac{\partial Q}{\partial v} \\ \frac{\partial Q^\intercal}{\partial v}\frac{\partial Q}{\partial u} & \frac{\partial Q^\intercal}{\partial v}\frac{\partial Q}{\partial v} \end{pmatrix} = \begin{pmatrix} E & F \\ F & G \end{pmatrix}. \tag{2.4}$$

Iš (2.4) paviršiaus lokalusis ilgio elementas dx tenkina sąlygą

$$dx^2 = Edu^2 + 2Fdudv + Gdv^2,$$

lokalusis ploto dA elementas lygus

$$dA = \sqrt{\det \mathbf{I}}dudv = \sqrt{EG - F^2}dudv.$$

Pagal šiuos fundamentalius dydžius galimi trys būdai – izometrinis, konforminis, autalinis, – kuriais lokaliai konstruojama paviršiaus Q išklotinė Q^* pagal atvaizdį f:

$$f : Q(u,v) \to Q^*(u,v).$$

4 apibrėžimas. Atvaizdis $f : Q(u,v) \to Q^*(u,v)$ vadinamas **izometriniu**, jei parametrinių paviršių $Q(u,v)$ ir $Q^*(u,v)$ pirmos fundamentaliosios formos sutampa:

$$\mathbf{I} = \mathbf{I}^*.$$

5 apibrėžimas. Atvaizdis $f : Q(u,v) \to Q^*(u,v)$ vadinamas **konforminiu**, jei parametrinių paviršių $Q(u,v)$ ir $Q^*(u,v)$ pirmos fundamentaliosios formos yra tiesiškai priklausomos, t. y. jei egzistuoja teigiamai apibrėžta skaliarinė funkcija $n(u,v)$, tenkinanti sąlygą:

$$\mathbf{I} = n(u,v)\mathbf{I}^*.$$

6 apibrėžimas. Atvaizdis $f : Q(u,v) \to Q^*(u,v)$ vadinamas **autaliniu**, jei parametrinių paviršių $Q(u,v)$ ir $Q^*(u,v)$ pirmos fundamentaliosios formos determinantai lygūs:

$$\det \mathbf{I} = \det \mathbf{I}^*.$$

Pagal 4, 5 ir 6 apibrėžimus, sudarant $T_\triangle$ paviršių išklotines $\overline{T}_\triangle$, skiriami 3 pagrindiniai paviršių parametrizavimo metodai:

1. Izometriniai, jei sudarant paviršiaus $T_\triangle$ išklotinę $\overline{T}_\triangle$ išlaikomi trikampių kraštinių ilgiai [62].

2. Konforminiai, jei sudarant paviršiaus $T_\triangle$ išklotinę $\overline{T}_\triangle$ išlaikomi trikampių kampai [85].

3. Autaliniai, jei sudarant paviršiaus $T_\triangle$ išklotinę $\overline{T}_\triangle$ išlaikomi trikampių plotai [110].

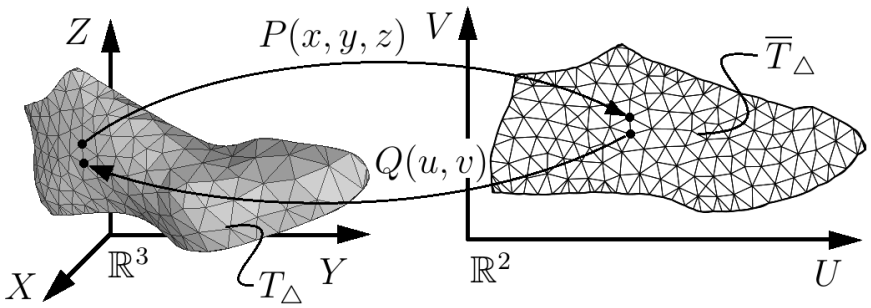
2.2.2 Skaitmeninių modelių paviršių išklotinės

3D skaitmeninio modelio paviršiaus išklotinės sudarymas (angl. *mesh parameterization*) apibrėžiamas kaip abipusio sąryšio sukonstravimas tarp šio modelio grafo viršūnių koordinačių trimatėje erdvėje ir to paties grafo viršūnių koordinačių plokštumoje, cilindro ar sferos paviršiuje. Remiantis Maksvelo ir Tuto teorema [100], kiekvieną $T_\triangle$ paviršių, topologišką sferai, galima planariai atvaizduoti vienetiniame apskritime (angl. *barycentric mapping*). Šios teoremos išvada – kiekvieną $T_\triangle$ paviršių, topologišką plokštumai, galima suprojektuoti plokštumoje α, viduje uždaro kontūro, sudaryto iš $T_\triangle$ paviršiaus kraštinių viršūnių tako, taip, kad jokios dvi briaunos, priklausančios suprojektuotam paviršiui $\overline{T}_\triangle$, nesikirstų. Matematiškai toks projektavimas apibrėžiamas paviršiaus $T_\triangle$ parametrizavimu

$$f : T_\triangle \leftrightarrow \overline{T}_\triangle, T_\triangle \subset \mathbb{R}^3, \overline{T}_\triangle \subset \mathbb{R}^2,$$

kur egzistuoja ryšiai $P(x,y,z) : T_\triangle \to \overline{T}_\triangle$ ir $Q(u,v) : \overline{T}_\triangle \to T_\triangle$ (2.5 paveikslas).

Tolesniuose skyriuose apžvelgsime šio parametrizavimo uždavinio sprendimo būdus.



2.5 paveikslas. Trikampių tinklo paviršiaus išklotinės sudarymo pavyzdys

2.2.3 Izometrinis paviršių parametrizavimas

Izometrinių paviršių parametrizavimo metodų grupei priklauso metodai, pagal kuriuos sudarant paviršių fragmentų išklotines kiek galima išlaikomi trikampių kraštinių ilgiai. Vienas populiariausių paviršiaus išklotinės sudarymo algoritmų – ARAP (*angl. as rigid as possible*) [62], kurį realizuojant sudaromos išklotinės kiek galima išlaikant $T_\triangle$ paviršiaus kiekvieno trikampio kraštinių ilgius arba kitaip – $T_\triangle$ paviršiaus standumą. ARAP algoritmo tikslo funkcija:

$$E(u, \mathbf{L}) = \sum_{t=1}^{T} A_t \|\mathbf{J}_t(u) - \mathbf{L}_t\|_F^2 = \sum_{t=1}^{T} A_t \mathrm{tr}((\mathbf{J}_t(u) - \mathbf{L}_t)^\intercal (\mathbf{J}_t(u) - \mathbf{L}_t)),$$

arba

$$E(u, \mathbf{L})) = \frac{1}{2} \sum_{t=1}^{T} \sum_{i=0}^{2} \cot(\theta_{t,i}) \|(u_{t,i} - u_{t,i+1}) - \mathbf{L}_t(x_{t,i} - x_{t,i+1})\|^2,$$

kur $x_t = \{x_{t,0}, x_{t,1}, x_{t,2}\}$ – 3D objekto paviršiaus trikampiai, $u_t = \{u_{t,0}, u_{t,1}, u_{t,2}\}$ – 3D objekto paviršiaus išklotinės trikampiai, $\theta_{t,i}$ – kampas prieš briauną $\{x_{t,i}, x_{t,i+1}\}$, kurios viršūnių indeksai x_t redukuoti moduliu 2, $\mathbf{L}_t$ – 2×2 posūkio matrica tarp trikampių x_t ir u_t, A_t – t-ojo trikampio plotas, $t = 1, \ldots, T$, $\|\mathbf{M}\|$ – 2×2 matricos $\mathbf{M} = (m_{i,j})$ Frobenijaus norma

$$\sqrt{m_{1,1}^2 + m_{1,2}^2 + m_{2,1}^2 + m_{2,2}^2},$$

$\mathrm{tr}(\mathbf{N})$ – 2×2 matricos $\mathbf{N} = (n_{i,j})$ pėdsakas

$$n_{1,1} + n_{2,2},$$

$\mathbf{J}_t(u)$ – t-ojo trikampio 2×2 Jakobio transformacijos matrica

$$\mathbf{J}_t(u) = \sum_{i=0}^{2} \cot(\theta_{t,i})(u_{t,i} - u_{t,i+1})(x_{t,i} - x_{t,i+1})^\intercal.$$

ARAP algoritmas minimizuoja tikslo funkciją $E(u, \mathbf{L})$ pakaitomis vykdydamas lokalią ir globalią optimizavimo fazes. Lokalioje optimizavimo fazėje kiekviena 2×2 Jakobio transformacijos matrica $\mathbf{J}_t(u)$, $t = 1, \ldots, T$ faktorizuojama į singuliariąją matricų dekompoziciją

$$\mathbf{J}_t(u) = \mathbf{U}_t \mathbf{\Sigma}_t \mathbf{V}_t^\intercal$$

ir apskaičiuojama posūkio matrica $\mathbf{L}_t = \mathbf{U}_t \mathbf{V}_t^\intercal$. Globalioje optimizavimo fazėje tikslo funkcijos

$$E(u, \mathbf{L}) = \frac{1}{2} \sum_{t=1}^{T} \sum_{i=0}^{2} \cot(\theta_{t,i}) \|(u_{t,i} - u_{t,i+1}) - \mathbf{L}_t(x_{t,i} - x_{t,i+1})\|^2 =$$

$$= \frac{1}{2} \sum_{i,j \in E^*} \cot(\hat{\theta}_{i,j}) \|(u_i - u_j) - \mathbf{L}_{t(i,j)}(x_i - x_j)\|^2$$

gradientą prilyginus nuliui gaunama Posono tiesinių lygčių sistema

$$\sum_{j \in N(i)} \cot(\hat{\theta}_{i,j} + \hat{\theta}_{j,i})(u_i - u_j) = \sum_{j \in N(i)} \cot(\hat{\theta}_{i,j} \mathbf{L}_{t(i,j)} + \hat{\theta}_{j,i} \mathbf{L}_{t(j,i)})(x_i - x_j), \; i = 1, \ldots, n.$$

Čia E^* – ribinių įvesties trikampių tinklo paviršiaus briaunų aibė, u_i ir x_i – i-osios viršūnės koordinatės, $t(i, j)$ – trikampis, kuriam priklauso ribinė kraštinė, $\hat{\theta}_{i,j}$ – trikampio $t(i, j)$ kampas prieš $\{i, j\}$ briauną. Šios lygčių sistemos sprendiniai $u_t = \{u_{t,0}, u_{t,1}, u_{t,2}\}$ – paviršiaus išklotinės trikampių tinklo viršūnių koordinatės. Pakaitomis vykdant lokalią ir globalią optimizavimo fazes, $E(u, \mathbf{L})$ reikšmė artėja į 0. Praktikoje užtenka $E(u, \mathbf{L})$ mažinti iki 0,001 kol galiausiai gaunama paviršiaus išklotinė.

Disertacijos prieduose pateiktas ARAP algoritmo pirminis programos tekstas, skirtas *Maple*[7] programavimo aplinkai.

2.2.4 Konforminis paviršių parametrizavimas

Šiame poskyryje apžvelgiami konforminiai paviršių parametrizavimo metodai. ABF algoritmas (angl. *angle based flattening*), pasiūlytas 2001 m. *Allos Sheffer, Erico de Sturlerio* [86], priskiriamas konforminių paviršių išklotinių sudarymo klasei. ABF minimizuoja tikslo funkciją F:

$$F(\alpha, \lambda_1, \lambda_2, \lambda_3) = \sum_{t \in T} \Big(\sum_{k=1}^{3} \frac{1}{w_k^t} (\alpha_k^t - \beta_k^t) + \lambda_1^t C_1(t) \Big) + \sum_{v \in V} \sum_{i=2}^{3} \lambda_i^v C_i(v),$$

[7] Matematinių funkcijų paketas ir kompiuterinio modeliavimo programinė įranga *Maple*. Internetinė prieiga: http://www.maplesoft.com.

kur $T \in T_\triangle$ paviršiui priklausančių trikampių aibė, V – šio paviršiaus viršūnių aibė, α_k^t nežinomi išklotinės trikampių kampai, $\lambda_1, \lambda_2, \lambda_3$ – nežinomi Lagranžo daugikliai, β_k^t yra duoti pradiniai paviršiaus trikampių kampai ir $w_k^t = \frac{1}{{\beta_k^t}^2}$ yra parinkti svoriai, kurie atspindi labiau santykinius nei absoliučius kampų iškraipymus.

Tikslo funkcija $F(\alpha, \lambda_1, \lambda_2, \lambda_3)$ minimizuojama laikantis trijų sąlygų:

1. Kiekvieno plokštumos trikampio kampų suma lygi π:

$$\forall t \in T,\ C_1(t) = \alpha_1^t + \alpha_2^t + \alpha_3^t - \pi.$$

2. Kiekvienas plokštumos pilnutinis kampas lygus 2π:

$$\forall v \in V_{int},\ C_2(v) = \sum_{(t,k)\in v^*} \alpha_k^t - 2\pi,$$

   kur V_{int} – vidinių viršūnių aibė, v^* – aibė kampų, incidenčių viršūnei v.

3. Gretimi plokštumos trikampiai turi vienodo ilgio bendrą kraštinę (ši sąlyga gaunama naudojantis sinusų teorema):

$$\forall v \in V_{int},\ C_3(v) = \prod_{(t,k)\in v^*} \sin \alpha_{k\oplus 1}^t - \prod_{(t,k)\in v^*} \sin \alpha_{k\ominus 1}^t,$$

   čia $k \oplus 1$ ir $k \ominus 1$ atitinkamai žymi tolesnį ir buvusį trikampio kampą laikrodžio rodyklės kryptimi, v – vidinė išklotinės viršūnė.

Pažymėkime $x = \{\alpha, \lambda\}$, x_0 – pradinį sprendinį. Siekiant minimizuoti $F(x)$, taikomas Niutono metodas [91], kurio pseudokodas yra pateiktas 1 algoritme.

1 algoritmas. Niutono metodo taikymas ABF algoritmui

Įvesties duomenys : F, x_0

Išvesties duomenys: x

```
x ← x_0;
while ‖∇F(x)‖ > ε do
    δ ← solve {∇²F(x)δ = −∇F(x)};
    x ← x + δ;
end
```

Šio algoritmo kiekvienos iteracijos rezultatas – vis tikslesnė paviršiaus išklotinė kiekvieno plokštumos trikampio kampų atžvilgiu. Trečioje 1 algoritmo eilutėje sprendžiama tiesinių lygčių sistema

$$\begin{pmatrix} \mathbf{\Lambda} & \mathbf{J}^\intercal \\ \mathbf{J} & 0 \end{pmatrix} \begin{pmatrix} \delta_\alpha \\ \delta_\lambda \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \tag{2.5}$$

čia matricos:

$$\mathbf{\Lambda} = \operatorname{diag}\left(\frac{2}{w_k^t}\right), \ \mathbf{J} = \left(\frac{\partial^2 F}{\partial \lambda_i \partial \alpha_k^t}\right),$$

vektoriai:

$$b_1 = -\nabla_\alpha F, \ b_2 = -\nabla_\lambda F$$

ir nežinomi vektoriai $\delta_\alpha, \delta_\lambda$. Tiesinių lygčių sistemą $\nabla^2 F(x)\delta = -\nabla F(x)$ ABF algoritmo autoriai pasiūlė spręsti naudojant SuperLU [29] tiesinių lygčių sistemos sprendimo įrankį, skirtą retosioms matricoms. 2005 m. pasiūlyta išskaidyti tiesinių lygčių sistemos $\nabla^2 F(x)\delta = -\nabla F(x)$ matricas $\mathbf{\Lambda}$ ir $\mathbf{J}$ supaprastinant ABF algoritmo atliekamus skaičiavimus. Naujasis algoritmas, pavadintas ABF++ [85], grąžina tą patį rezultatą (paviršiaus išklotinę) per trumpesnį laiką atliekant algebrinius pertvarkymus, kuriais supaprastinama (2.5) lygčių sistema. Ją galima perrašyti:

$$\begin{cases} \mathbf{\Lambda}\delta_\alpha + \mathbf{J}^\intercal\delta_\lambda = b_1, \\ \mathbf{J}\delta_\alpha = b_2. \end{cases} \tag{2.6}$$

Apskaičiuokime nežinomų Lagranžo daugiklių vektorių δ_λ bei išreikškime vektorių δ_α. Iš kairės padauginus (2.6) sistemos pirmąją lygtį iš $\mathbf{J}\mathbf{\Lambda}^{-1}$ ir atėmus iš jos antrąją gauname

$$\mathbf{J}\mathbf{\Lambda}^{-1}\mathbf{J}^\intercal\delta_\lambda = \mathbf{J}\mathbf{\Lambda}^{-1}b_1 - b_2. \tag{2.7}$$

Pažymėkime $\lambda = \{\lambda_1, \lambda_2, \lambda_3\}$. Naudojant šią išraišką 1 algoritmą galima perrašyti į 2 algoritmą.

2 algoritmas. ABF++ algoritmas: pirmasis matricos atskyrimas

Įvesties duomenys : F, x_0

Išvesties duomenys: $x = \{\alpha, \lambda\}$

$x \leftarrow x_0$;
while $\|\nabla F(x)\| > \epsilon$ **do**
  $b, \mathbf{J}, \mathbf{\Lambda} \leftarrow F(x)$ pagal (2.5);
  $\delta_\lambda \leftarrow$ **solve** $\left\{\mathbf{J}\mathbf{\Lambda}^{-1}\mathbf{J}^\intercal\delta_\lambda = \mathbf{J}\mathbf{\Lambda}^{-1}b_1 - b_2\right\}$;
  $\delta_\alpha \leftarrow \mathbf{\Lambda}^{-1}\left(b_1 - \mathbf{J}^\intercal\delta_\lambda\right)$;
  $\lambda \leftarrow \lambda + \delta_\lambda$;
  $\alpha \leftarrow \alpha + \delta_\alpha$;
end

5-oje 2 algoritmo eilutėje δ_α išreikštas iš (2.6) sistemos pirmosios lygties. Įvertinę matricos $\mathbf{J}$ struktūrą, galime dar sumažinti 2 algoritmo realizacijos metu atliekamus skaičiavimus. Išskaidykime matricą $\mathbf{J}$ į $\mathbf{J_1}$ ir $\mathbf{J_2}$ matricas, kurių matmenys atitinkamai lygūs $|T| \times 3|T|$ ir $2|V_{int}| \times 3|T|$:

$$\mathbf{J} = \begin{pmatrix} \mathbf{J_1} \\ \mathbf{J_2} \end{pmatrix},$$

čia $\mathbf{J_1}$ – tikslo funkcijos antros eilės išvestinių matrica, turinti struktūrą

$$\mathbf{J_1} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 1 \end{pmatrix}.$$

Iš (2.7) $\mathbf{J}\mathbf{\Lambda}^{-1}\mathbf{J}^\intercal$ išraišką galime išskaidyti į dekompoziciją

$$\mathbf{J}\mathbf{\Lambda}^{-1}\mathbf{J}^\intercal = \begin{pmatrix} \mathbf{\Lambda}^* & \mathbf{J}^{*\intercal} \\ \mathbf{J}^* & \mathbf{J}^{**} \end{pmatrix} = \begin{pmatrix} \mathbf{J_1}\mathbf{\Lambda}^{-1}\mathbf{J_1}^\intercal & \mathbf{J_1}\mathbf{\Lambda}^{-1}\mathbf{J_2}^\intercal \\ \mathbf{J_2}\mathbf{\Lambda}^{-1}\mathbf{J_1}^\intercal & \mathbf{J_2}\mathbf{\Lambda}^{-1}\mathbf{J_2}^\intercal \end{pmatrix}.$$

Pažymėkime

$$b^* = \mathbf{J}\mathbf{\Lambda}^{-1}b_1 - b_2 = \begin{pmatrix} b_1^* \\ b_2^* \end{pmatrix}.$$

Tada (2.7) įgyja išraišką $\mathbf{J}\mathbf{\Lambda}^{-1}\mathbf{J}^\intercal\delta_\alpha = b^*$ arba

$$\begin{pmatrix} \mathbf{\Lambda}^* & \mathbf{J}^{*\intercal} \\ \mathbf{J}^* & \mathbf{J}^{**} \end{pmatrix} \begin{pmatrix} \delta_{\lambda_1} \\ \delta_{\lambda_2} \end{pmatrix} = \begin{pmatrix} b_1^* \\ b_2^* \end{pmatrix}.$$

Iš čia

$$\begin{cases} \mathbf{\Lambda}^* \delta_{\lambda_1} + \mathbf{J}^{*\intercal} \delta_{\lambda_2} = b_1^*, \\ \mathbf{J}^* \delta_{\lambda_1} + \mathbf{J}^{**} \delta_{\lambda_2} = b_2^*. \end{cases} \tag{2.8}$$

Kadangi $\mathbf{J_1}$ yra ortogonalioji matrica ir $\mathbf{\Lambda}^{-1}$ – diagonalinė matrica, tai ir $\mathbf{\Lambda}^* = \mathbf{J_1}\mathbf{\Lambda}^{-1}\mathbf{J_1}^\intercal$ – diagonalinė matrica, todėl padauginę (2.8) sistemos pirmąją lygtį iš $\mathbf{J}^*\mathbf{\Lambda}^{*-1}$ ir atėmę antrąją gauname

$$\big(\mathbf{J}^*\mathbf{\Lambda}^{*-1}\mathbf{J}^{*\intercal} - \mathbf{J}^{**}\big)\delta_{\lambda_2} = \mathbf{J}^*\mathbf{\Lambda}^{*-1} b_1^* - b_2^*. \tag{2.9}$$

Galiausiai naudojant (2.9) dekompoziciją sprendiniui δ_{λ_2} rasti ABF++ algoritmo pagrindinę dalį galima perrašyti į 3 algoritmą.

3 algoritmas. ABF++ algoritmas: antrasis matricos atskyrimas

Įvesties duomenys : F, x_0

Išvesties duomenys: $x = \{\alpha, \lambda\}$

$x \leftarrow x_0$;
while $\|\nabla F(x)\| > \epsilon$ **do**
  $b, \mathbf{J}, \mathbf{\Lambda} \leftarrow F(x)$ pagal (2.5);
  $\delta_{\lambda_2} \leftarrow$ **solve** $\big\{\big(\mathbf{J}^*\mathbf{\Lambda}^{*-1}\mathbf{J}^{*\intercal} - \mathbf{J}^{**}\big)\delta_{\lambda_2} = \mathbf{J}^*\mathbf{\Lambda}^{*-1} b_1^* - b_2^*\}$;
  $\delta_{\lambda_1} \leftarrow \mathbf{\Lambda}^{*-1}\big(b_1^* - \mathbf{J}^{*\intercal}\delta_{\lambda_2}\big)$;
  $\delta_\alpha \leftarrow \mathbf{\Lambda}^{-1}\big(b_1 - \mathbf{J}^\intercal \delta_\lambda\big)$;
  $\lambda_1 \leftarrow \lambda_1 + \delta_{\lambda_1}$;
  $\lambda_2 \leftarrow \lambda_2 + \delta_{\lambda_2}$;
  $\alpha \leftarrow \alpha + \delta_\alpha$;
end

5-oje 3 algoritmo eilutėje δ_{λ_1} išreikštas iš (2.8) sistemos pirmosios lygties. Realizavus 3 algoritmą apskaičiuojami paviršiaus išklotinei $\overline{T}_\triangle$ priklausančių trikampių kampai. Šios išklotinės viršūnių koordinates galima apskaičiuoti fiksuojant bet kurių dviejų gretimų viršūnių koordinates.

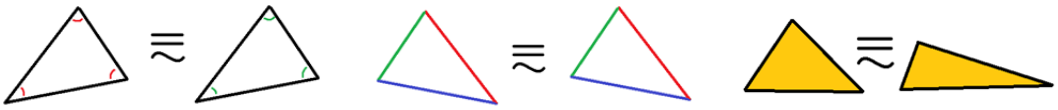
LSCM (angl. *least-squares conformal mapping*) paviršiaus išklotinės sudarymo algoritmas, pasiūlytas autorių *B. Lévy'o, S. Petitjeano, N. Ray'aus ir J. Maillo-to* [58], priskiriamas konforminių $T_\triangle$ paviršių išklotinių sudarymo klasei, t. y. išlaikant panašius trikampius. LSCM minimizuoja tikslo funkciją – konforminę išklotinės $\overline{T}_\triangle$ energiją $\varepsilon_C(S)$:

$$\varepsilon_C(S) = \int_S \|\nabla v - (\nabla u)^\perp\| = \int_S \Big|\frac{\partial X(t)}{\partial v} + i\frac{\partial X(t)}{\partial u}\Big|^2 dt,$$

čia $S \in T_{\triangle}$ – pradinio paviršiaus sritis, $\nabla v = (\nabla u)^{\perp}$ – Koši ir Rymano lygčių sistema ir $X : \mathbb{R}^3 \to \mathbb{C};\ (x, y, z) \mapsto u + iv$ – atvirkštinis parametrizavimas. Parametrinėje erdvėje konforminė energija ε_C atitinka dviejų kintamųjų (u_k, v_k) funkciją. Plokštumoje šias koordinates galima apskaičiuoti naudojant greičiausio nusileidimo gradientinį metodą [79].

2.2.5 Autalinis paviršių parametrizavimas

Sudarant skaitmeninių modelių paviršių išklotines autalinis parametrizavimo metodas nėra naudojamas, nes lokaliai išplokštintoje srityje, išlaikant tą patį plotą, trikampis gali įgyti bet kokią formą (2.6c paveikslas).



(a) Konforminis **(b)** Izometrinis **(c)** Autalinis

2.6 paveikslas. Paviršiaus parametrizavimo metodai

Autalinis paviršių parametrizavimo metodas naudojamas žinant parametrinę paviršiaus lygtį, pavyzdžiui, sferos paviršiaus. Nagrinėjant 3D skaitmeninius modelius parametrinę paviršiaus išraišką reikia sukonstruoti aproksimuojant 3D skaitmeninio modelio taškų debesį, tačiau šioje disertacijoje nebuvo nagrinėjami parametrinių paviršių sudarymo metodai iš trimačio taškų debesies, todėl autalinis paviršių parametrizavimo metodas ir nėra apžvelgtas. Kita vertus, autaliniu parametrizavimo metodu sudarant paviršiaus išklotinę išlaikomas pradinio segmento plotas (2.6c paveikslas), o sudarant kurpalių lekalus svarbiausia išlaikyti paviršiaus standumą (2.6a, 2.6b paveikslai), t. y. naudoti konforminį ar izometrinį parametrizavimo metodą.

Kitame poskyryje apžvelgsime kurpalių paviršių segmentavimo taisykles, pagal kurias sudaromi segmentai paruošiami išplokštinti.

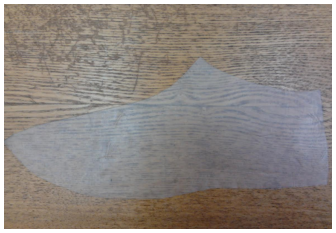
2.3 Rankinis kurpalių paviršių segmentavimas ir gamybinių brėžinių sudarymas

Šiame poskyryje nagrinėsime rankinį kurpalių paviršių segmentavimą, kuris naudojamas ortopedijoje gaminant vienetinius produktus.

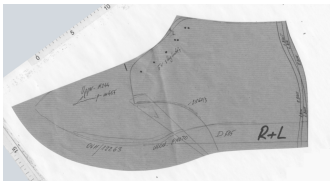
(a) Kurpalio paviršiaus segmentavimas

(b) Kurpalio paviršius padengiamas pakaitinta vaško folija

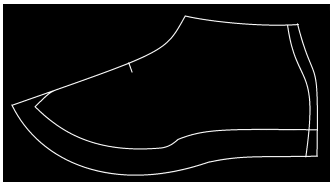
(c) Pagal segmentų linijas vaško folija supjaustoma ir nuimama



(d) Vaško folija vėl pakaitinama ir išplokštinama



(e) Pagal vaško folijos kontūrą sudaromi gamybiniai brėžiniai



(f) Gamybinių brėžinių suskaitmeninimas

2.7 paveikslas. Kurpalių lekalų sudarymas naudojant gamybinę vaško foliją

Pagal ortopedijos ekspertų taisykles kurpalio paviršiaus dalijimas ir gamybinių brėžinių sudarymas atliekamas šiais žingsniais:

1. Pagal paviršiaus kreivumo linijas atskiriamas viršutinis ir apatinis kurpalio kontūrai. Apatinis kurpalio kontūras atitinka padą, viršutinis – bato viršų arba kojos pjūvį. Viršutinis ir apatinis kurpalių kontūrai sudaromi šalinant kurpalio padą ir viršų atsižvelgiant į paviršiuje esančią išlinkio kreivę, kuri žymi perėjimą į kurpalio šonus. Tada kurpalio dalijimas plokštuma atliekamas pagal šią taisyklę: kurpalyje pažymimas taškas, kuris atitinka antrojo kojos piršto vidurį. Per šį tašką paviršiumi brėžiama tiesi linija kurpalio viršutinės srities aritmetinio vidurkio taško link ir galutinis brėžimas vyksta labiausiai nutolusio kulno taško link (2.7a paveikslas).

2. Kurpalis dalijamas išilgai plokštuma per šiuos tris taškus:

a) Antrojo kojos piršto vidurį atitinkančio taško.

b) Labiausiai nutolusio kulno taško.

c) Viršutinio kontūro vidurio taško.

3. Apskaičiuojamas įtvirčio taškas (angl. *vamp point*):

   a) Apskaičiuojamas trumpiausias atstumas tarp tolimiausio kurpalio kulno taško ir artimiausio pėdos kontūro taško. Šis atstumas apskaičiuojamas naudojant virvę, kuri ištempiama kurpalio viršutinės dalies paviršiumi.

   b) Įtvirčio taškas pažymimas kurpalio priekinėje viršutinėje dalyje pridėjus trečdalį apskaičiuoto atstumo ant kurpalio ir jį išilgai dalijančios plokštumos sankirtos.

4. Naudojant gamybinę vaško foliją sudaromos dvi šoninių kurpalio paviršiaus segmentų išklotinės (2.7b, 2.7c, 2.7d paveikslai).

5. Šios dvi vaško folijos išklotinės sudedamos viena ant kitos sutapatinus įtvirčio taškus ir apskaičiuojamas suvidurkintas šių išklotinių priekinės dalies kontūras, o galinės (kulno) dalies kontūras nevidurkinamas. Galiausiai gamybiniuose brėžiniuose pridedama papildoma 2–3 cm odos užlaida apatinėje (pado) lekalo dalyje (2.7e, 2.7f paveikslai).

Sudaryti gamybiniai brėžiniai naudojami odai iškarpyti pagal bato dizainą (2.7e paveikslas). Šie brėžiniai taip pat suskaitmeninami (2.7f paveikslas).

Kitame poskyryje nagrinėsime 2×2 matricų skaidymą singuliariosiomis reikšmėmis. Pagal singuliariąsias matricų reikšmes apskaičiuojami paviršių deformacijos skaitiniai įverčiai, sudaromi optimizavimo uždaviniai trikampių tinklo paviršių transformuojant iš trimatės erdvės į plokštumą.

2.4 Matricų skaidymas singuliariosiomis reikšmėmis

Daugelio programavimo kalbų bibliotekose integruoti matricų skaidymo singuliariosiomis reikšmėmis algoritmai. Tačiau išsigimusiųjų matricų atvejais kai kurie

algoritmai gali grąžinti klaidingą rezultatą, įskaitant patį paprasčiausią 2×2 atvejį išsigimusiosioms matricoms. Šiuo atskiru atveju patogu naudoti išreikštines matricų skaidymo singuliariosiomis reikšmėmis formules atskiram 2×2 atvejui nenaudojant programų bibliotekų. Išreikštinės singuliariųjų reikšmių $\sigma_{1,1}$ ir $\sigma_{2,2}$ formulės taip pat labai naudingos skaičiuojant paviršių deformacijas, nes atliekami tik būtini skaičiavimai ir taupoma kompiuterio atmintis. Šiame skyriuje pateikiami darbo [6A] rezultatai, skirti 2×2 realiųjų matricų skaidymui singuliariosiomis reikšmėmis.

2.4.1 Singuliarioji matricų dekompozicija

Tiesinėje algebroje SVD (angl. *singular value decomposition*) yra vadinama faktorizacija $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\intercal$, kur $\mathbf{A}$ – pradinė matrica, $\mathbf{U}$ ir $\mathbf{V}$ – unitariosios matricos ir $\mathbf{\Sigma}$ diagonalinė matrica, kurios elementai neneigiami. Paprasčiausiu 2×2 atveju $\mathbf{A}$ tenkina sąlygą:

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} = \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{pmatrix} \begin{pmatrix} \sigma_{1,1} & 0 \\ 0 & \sigma_{2,2} \end{pmatrix} \begin{pmatrix} v_{1,1} & v_{2,1} \\ v_{1,2} & v_{2,2} \end{pmatrix}, \tag{2.10}$$

kai

$$\mathbf{U}\mathbf{U}^\intercal = \mathbf{V}\mathbf{V}^\intercal = \mathbf{I}, \quad \sigma_{1,1}, \sigma_{2,2} \geqslant 0. \tag{2.11}$$

2.4.2 Analitinis SVD kompozicijos sprendinys

J. Blinnas pasiūlė tiesioginį sprendimą [12] matricos $\mathbf{A}$ dekompozicijai

$$\begin{pmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{pmatrix} = \begin{pmatrix} \cos(\delta) & \sin(\delta) \\ -\sin(\delta) & \cos(\delta) \end{pmatrix} \begin{pmatrix} \sigma_{1,1} & 0 \\ 0 & \sigma_{2,2} \end{pmatrix} \begin{pmatrix} \cos(\gamma) & \sin(\gamma) \\ -\sin(\gamma) & \cos(\gamma) \end{pmatrix}.$$

Išskaidykime matricą $\mathbf{A}$ į dviejų matricų sumą įvedę naujų elementų žymėjimus $\hat{E}, \hat{F}, \hat{G}$ ir $\hat{H}$:

$$\begin{pmatrix} \hat{A} & \hat{B} \\ \hat{C} & \hat{D} \end{pmatrix} = \begin{pmatrix} \hat{E} & \hat{H} \\ -\hat{H} & \hat{E} \end{pmatrix} + \begin{pmatrix} \hat{F} & \hat{G} \\ \hat{G} & -\hat{F} \end{pmatrix}.$$

4 nežinomieji $\hat{E}, \hat{F}, \hat{G}, \hat{H}$ apskaičiuojami išsprendus 4 lygčių sistemą

$$\begin{cases} \hat{A} = \hat{E} + \hat{F}, \\ \hat{B} = \hat{H} + \hat{G}, \\ \hat{C} = -\hat{H} + \hat{G}, \\ \hat{D} = \hat{E} - \hat{F}. \end{cases}$$

Nežinomi matricos Σ elementai $\sigma_{1,1}$ ir $\sigma_{2,2}$ apskaičiuojami išsprendus lygčių sistemą

$$\begin{cases} \frac{\sigma_{1,1}+\sigma_{2,2}}{2} = \sqrt{\hat{E}^2 + \hat{H}^2}, \\ \frac{\sigma_{1,1}-\sigma_{2,2}}{2} = \sqrt{\hat{F}^2 + \hat{G}^2}. \end{cases} \tag{2.12}$$

Galiausiai nežinomieji γ ir δ apskaičiuojami išsprendus lygčių sistemą

$$\begin{cases} \gamma - \delta = \arctan(\hat{G}/\hat{F}), \\ \gamma + \delta = \arctan(\hat{H}/\hat{E}). \end{cases} \tag{2.13}$$

Šios formulės tinkamos apskaičiuoti 2×2 faktorizaciją $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\intercal$ visais atvejais, išskyrus, kai $\hat{G} = \hat{F} = 0$ ir $\hat{H} = \hat{E} = 0$. Be to, pagal (2.12) lygybes, $\sigma_{2,2}$ gali įgyti neigiamas reikšmes ir tada gaunama prieštara pradinei (2.11) sąlygai, todėl disertacijoje pasiūlytas analitinis SVD faktorizacijos sprendinys, kuris tenkina pradines (2.10, 2.11) sąlygas ir yra apibrėžtas visoms realiosioms matricoms $\mathbf{A}$. Remiantis *J. Blinno* pasiūlytomis (2.12) ir (2.13) formulėmis galima apskaičiuoti $\sigma_{1,1} \geqslant 0, \sigma_{2,2}$, ir γ reikšmes:

$$\sigma_{1,1} = \frac{1}{2}\left(\sqrt{(a_{1,1} - a_{2,2})^2 + (a_{1,2} + a_{2,1})^2} + \sqrt{(a_{1,1} + a_{2,2})^2 + (a_{1,2} - a_{2,1})^2}\right). \tag{2.14}$$

Skaičiuojant reikšmę $\sigma_{2,2}$ pagal pradinę neneigiamumo sąlygą (2.11) reikia pridėti modulį, todėl galutinė formulė

$$\sigma_{2,2} = \left|\sigma_{1,1} - \sqrt{(a_{1,1} - a_{2,2})^2 + (a_{1,2} + a_{2,1})^2}\right|. \tag{2.15}$$

Apskaičiuokime pirma $v_{2,1}$ reikšmę, nes $v_{1,1}$ yra teigiama ir negali vienareikšmiškai apibrėžti likusių matricos $\mathbf{V}$ reikšmių.

$$\begin{aligned} v_{2,1} = \sin(\gamma) &= \sin\left(\frac{1}{2}\left(\tan^{-1}\left(\frac{a_{1,2} + a_{2,1}}{a_{1,1} - a_{2,2}}\right) + \tan^{-1}\left(\frac{a_{1,2} - a_{2,1}}{a_{1,1} + a_{2,2}}\right)\right)\right) \\ &= \sin\left(\frac{1}{2}\left(\tan^{-1}\left(\frac{2\left(a_{1,1}a_{1,2} + a_{2,1}a_{2,2}\right)}{a_{1,1}^2 - a_{1,2}^2 + a_{2,1}^2 - a_{2,2}^2}\right)\right)\right), \end{aligned} \tag{2.16}$$

naudojant arktangentų sumos formulę

$$\tan^{-1}(x) + \tan^{-1}(y) = \tan^{-1}\left(\frac{x+y}{1-xy}\right).$$

Pašalinkime iš (2.16) trigonometrines funkcijas keisdami arctan į atan2. Tegu $\alpha = 2a_{1,1}a_{1,2} + 2a_{2,1}a_{2,2}$ ir $\beta = a_{1,1}^2 - a_{1,2}^2 + a_{2,1}^2 - a_{2,2}^2$. Tada

$$\sin\left(\frac{\operatorname{atan2}(\alpha,\beta)}{2}\right) = \frac{\sqrt{2}}{2}\operatorname{sgn}(\alpha)\sqrt{1 - \frac{\operatorname{sgn}(\beta)}{\sqrt{1+(\alpha/\beta)^2}}}$$

$$= \operatorname{sgn}(\alpha)\sqrt{\frac{1}{2} - \frac{|\beta|\operatorname{sgn}(\beta)}{2\sqrt{\alpha^2+\beta^2}}} = \operatorname{sgn}(\alpha)\sqrt{\frac{\sigma_{1,1}^2 - \sigma_{2,2}^2 - \beta}{2(\sigma_{1,1}^2 - \sigma_{2,2}^2)}},$$

kur

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan(y/x), & x > 0, \\ \arctan(y/x) + \pi, & y \geqslant 0, \ x < 0, \\ \arctan(y/x) - \pi, & y < 0, \ x < 0, \\ +\frac{\pi}{2}, & y > 0, \ x = 0, \\ -\frac{\pi}{2}, & y < 0, \ x = 0, \\ \text{neapibrėžtas}, & y = 0, \ x = 0. \end{cases}$$

Nesunku patikrinti, kad

$$\sqrt{\alpha^2+\beta^2} = \sigma_{1,1}^2 - \sigma_{2,2}^2.$$

Buvusios formulės apibrėžtos, kai $\sigma_{1,1} > \sigma_{2,2}$. Tačiau lengvai galime patikrinti, kad nelygybė $\sigma_{1,1} < \sigma_{2,2}$ neįmanoma. Kai $\sigma_{1,1} = \sigma_{2,2} = 0$, turime nulinę matricą $\mathbf{A}$, o $\mathbf{U}$ ir $\mathbf{V}$ galima priskirti vienetinei matricai $\mathbf{I}$.

$$v_{1,1} = \sqrt{1 - v_{2,1}^2} = \sqrt{1 - \left(\operatorname{sgn}(\alpha)\sqrt{\frac{\sigma_{1,1}^2 - \sigma_{2,2}^2 - \beta}{2(\sigma_{1,1}^2 - \sigma_{2,2}^2)}}\right)^2} = \sqrt{\frac{\sigma_{1,1}^2 - a_{1,2}^2 - a_{2,2}^2}{\sigma_{1,1}^2 - \sigma_{2,2}^2}},$$

nes $\sigma_{1,1}^2 + \sigma_{2,2}^2 = a_{1,1}^2 + a_{1,2}^2 + a_{2,1}^2 + a_{2,2}^2$. Čia

$$\operatorname{sgn}(\alpha) = \begin{cases} 1, & \alpha > 0, \\ 0, & \alpha = 0, \\ -1, & \alpha < 0. \end{cases}$$

Galų gale

$$v_{1,1} = \begin{cases} \sqrt{(\sigma_{1,1}^2 - a_{1,2}^2 - a_{2,2}^2)/(\sigma_{1,1}^2 - \sigma_{2,2}^2)}, & \sigma_{1,1} > \sigma_{2,2}, \\ 1, & \text{kitu atveju.} \end{cases}$$

Pagal $\operatorname{sgn}(\alpha)$ reikšmę matricos $\mathbf{V}$ reikšmės lygios

$$v_{2,1} = \begin{cases} -\sqrt{1-v_{1,1}^2}, & a_{1,1}a_{1,2} + a_{2,1}a_{2,2} < 0, \\ \sqrt{1-v_{1,1}^2}, & \text{kitu atveju,} \end{cases}$$

tada $v_{1,2} = -v_{2,1}, \quad v_{2,2} = v_{1,1}$.

Belieka išspręsti (2.10), (2.11) lygtis. Jei $\sigma_{1,1} = 0$, tai $\mathbf{A}$ yra nulinė matrica, o $\mathbf{U}$ turime priskirti vienetinei matricai $\mathbf{I}$:

$$u_{1,1} = \begin{cases} (a_{1,1}v_{1,1} + a_{1,2}v_{2,1})/\sigma_{1,1}, & \sigma_{1,1} \neq 0, \\ 1, & \text{kitu atveju,} \end{cases}$$

$$u_{2,1} = \begin{cases} (a_{2,1}v_{1,1} + a_{2,2}v_{2,1})/\sigma_{1,1}, & \sigma_{1,1} \neq 0, \\ 0, & \text{kitu atveju.} \end{cases}$$

Jei $\sigma_{2,2} = 0$, tai $\mathbf{A}$ yra išsigimusioji matrica. Kad $\mathbf{U}$ būtų unitarioji, turime priskirti $u_{1,2} = -u_{2,1}$ ir $u_{2,2} = u_{1,1}$:

$$u_{1,2} = \begin{cases} (a_{1,1}v_{1,2} + a_{1,2}v_{2,2})/\sigma_{2,2}, & \sigma_{2,2} \neq 0, \\ -u_{2,1}, & \text{kitu atveju,} \end{cases}$$

$$u_{2,2} = \begin{cases} (a_{2,1}v_{1,2} + a_{2,2}v_{2,2})/\sigma_{2,2}, & \sigma_{2,2} \neq 0, \\ u_{1,1}, & \text{kitu atveju.} \end{cases}$$

2.4.3 Išvesto sprendinio eksperimentinis patikrinimas

Siekiant eksperimentiškai patikrinti, ar išvestas SVD 2×2 analitinis sprendinys yra teisingas visais atvejais, t. y. ar tenkinamos (2.10), (2.11) sąlygos, buvo atsitiktinai sugeneruota 10 milijonų realiųjų matricų $\mathbf{M}^i : m_{1,1}^i, m_{1,2}^i, m_{2,1}^i, m_{2,2}^i \in (-1, 1),\ i = 1, \ldots, 10^7$ ir atliktas daugiau nei 10 milijonų sveikųjų matricų visiškas perrinkimas

$$\mathbf{N}^i : n_{1,1}^i, n_{1,2}^i, n_{2,1}^i, n_{2,2}^i \in [-28, 28],\ i = 1, \ldots, 57^4.$$

Tada šioms matricoms pagal 4 algoritmą apskaičiuotos singuliariosios matricos bei 6 sumos:

$$\sum_{i=1}^{10^7} \left(\left| \sigma_{1,1}^i u_{1,1}^i v_{1,1}^i + \sigma_{2,2}^i u_{1,2}^i v_{2,1}^i - m_{1,1}^i \right| + \left| \sigma_{1,1}^i u_{1,1}^i v_{1,2}^i + \sigma_{2,2}^i u_{1,2}^i v_{2,2}^i - m_{1,2}^i \right| +$$

$$+\left|\sigma^i_{1,1}u^i_{2,1}v^i_{1,1}+\sigma^i_{2,2}u^i_{2,2}v^i_{2,1}-m^i_{2,1}\right|+\left|\sigma^i_{1,1}u^i_{2,1}v^i_{1,2}+\sigma^i_{2,2}u^i_{2,2}v^i_{2,2}-m^i_{2,2}\right|\Bigg),$$

$$\sum_{i=1}^{10^7}\Bigg((u^i_{1,1})^2+(u^i_{1,2})^2-1+2\left|u^i_{1,1}u^i_{2,1}+u^i_{1,2}u^i_{2,2}\right|+(u^i_{2,1})^2+(u^i_{2,2})^2-1\Bigg),$$

$$\sum_{i=1}^{10^7}\Bigg((v^i_{1,1})^2+(v^i_{2,1})^2-1+2\left|v^i_{1,1}v^i_{1,2}+v^i_{2,1}v^i_{2,2}\right|+(v^i_{1,2})^2+(v^i_{2,2})^2-1\Bigg),$$

$$\sum_{i=1}^{57^4}\Bigg(\left|\sigma^i_{1,1}u^i_{1,1}v^i_{1,1}+\sigma^i_{2,2}u^i_{1,2}v^i_{2,1}-n^i_{1,1}\right|+\left|\sigma^i_{1,1}u^i_{1,1}v^i_{1,2}+\sigma^i_{2,2}u^i_{1,2}v^i_{2,2}-n^i_{1,2}\right|+$$

$$+\left|\sigma^i_{1,1}u^i_{2,1}v^i_{1,1}+\sigma^i_{2,2}u^i_{2,2}v^i_{2,1}-n^i_{2,1}\right|+\left|\sigma^i_{1,1}u^i_{2,1}v^i_{1,2}+\sigma^i_{2,2}u^i_{2,2}v^i_{2,2}-n^i_{2,2}\right|\Bigg),$$

$$\sum_{i=1}^{57^4}\Bigg((u^i_{1,1})^2+(u^i_{1,2})^2-1+2\left|u^i_{1,1}u^i_{2,1}+u^i_{1,2}u^i_{2,2}\right|+(u^i_{2,1})^2+(u^i_{2,2})^2-1\Bigg),$$

$$\sum_{i=1}^{57^4}\Bigg((v^i_{1,1})^2+(v^i_{2,1})^2-1+2\left|v^i_{1,1}v^i_{1,2}+v^i_{2,1}v^i_{2,2}\right|+(v^i_{1,2})^2+(v^i_{2,2})^2-1\Bigg).$$

Kadangi visos šios 6 sumos buvo lygios 0, vadinasi, visais atvejais 4 algoritmo rezultatas $\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{V}^\intercal$ tenkino (2.10), (2.11) sąlygas.

4 algoritmas. 2×2 matricų skaidymas singuliariosiomis reikšmėmis

Įvesties duomenys : A

Išvesties duomenys: $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$

$\sigma_{1,2}, \sigma_{2,1} \leftarrow 0$;

$\sigma_{1,1} \leftarrow \left(\sqrt{(a_{1,1}-a_{2,2})^2+(a_{1,2}+a_{2,1})^2}+\sqrt{(a_{1,1}+a_{2,2})^2+(a_{1,2}-a_{2,1})^2}\right)/2$;

$\sigma_{2,2} \leftarrow \left|\sigma_{1,1}-\sqrt{(a_{1,1}-a_{2,2})^2+(a_{1,2}+a_{2,1})^2}\right|$;

if $\sigma_{1,1} > \sigma_{2,2}$ **then** $v_{1,1} \leftarrow \sqrt{(\sigma^2_{1,1}-a^2_{1,2}-a^2_{2,2})/(\sigma^2_{1,1}-\sigma^2_{2,2})}$ **else** $v_{1,1} \leftarrow 1$ **end**;

if $a_{1,1}a_{1,2} < -a_{2,1}a_{2,2}$ **then** $v_{1,2} \leftarrow \sqrt{1-v^2_{1,1}}$ **else** $v_{1,2} \leftarrow -\sqrt{1-v^2_{1,1}}$ **end**;

$v_{2,1} \leftarrow -v_{1,2}$, $v_{2,2} \leftarrow v_{1,1}$;

if $\sigma_{1,1} \neq 0$ **then** $u_{1,1} \leftarrow (a_{1,1}v_{1,1}+a_{1,2}v_{2,1})/\sigma_{1,1}$ **else** $u_{1,1} \leftarrow 1$ **end**;

if $\sigma_{1,1} \neq 0$ **then** $u_{2,1} \leftarrow (a_{2,1}v_{1,1}+a_{2,2}v_{2,1})/\sigma_{1,1}$ **else** $u_{2,1} \leftarrow 0$ **end**;

if $\sigma_{2,2} \neq 0$ **then** $u_{1,2} \leftarrow (a_{1,1}v_{1,2}+a_{1,2}v_{2,2})/\sigma_{2,2}$ **else** $u_{1,2} \leftarrow -u_{2,1}$ **end**;

if $\sigma_{2,2} \neq 0$ **then** $u_{2,2} \leftarrow (a_{2,1}v_{1,2}+a_{2,2}v_{2,2})/\sigma_{2,2}$ **else** $u_{1,1} \leftarrow u_{2,2}$ **end**;

Kitame poskyryje nagrinėsime singuliariųjų matricos $\mathbf{\Sigma}$ reikšmių $\sigma_{1,1}, \sigma_{2,2}$ taikymą paviršių standumo ir panašumo deformacijos skaitiniams įverčiams apskaičiuoti.

2.5 Paviršiaus deformacijos įverčiai

Paviršiaus deformacijos įverčių taikymo tikslas – rasti įvertį, kurio skaitinės vertės dydis parodo deformacijos dydį. $T_\triangle$ paviršių išklotinių $\overline{T}_\triangle$ atveju reikia atskirai įvertinti kiekvieno trikampio deformacijos dydį ir apskaičiuoti visų tokių dydžių sumą.

Šiame skyriuje apžvelgsime darbe [62] pasiūlytą matematinį metodą, kuris skirtas sukonstruoti trikampių kovariacijos matricai, pagal kurią apskaičiuojamos paviršiaus deformacijos. Tegu turime 2 trikampius

$$\triangle ABC \in T_\triangle, \quad \triangle A^\alpha B^\alpha C^\alpha \in \overline{T}_\triangle,$$

kurių viršūnių koordinatės

$$A(a_1, a_2, a_3),\ B(b_1, b_2, b_3),\ C(c_1, c_2, c_3),$$

$$A^\alpha(a_1^\alpha, a_2^\alpha),\ B^\alpha(b_1^\alpha, b_2^\alpha),\ C^\alpha(c_1^\alpha, c_2^\alpha).$$

Pažymėkime $\triangle ABC$ kampų kotangentų reikšmes

$$q_1 = \cot(\angle CAB),\ q_2 = \cot(\angle ABC),\ q_3 = \cot(\angle ACB).$$

Pažymėkime suprojektuoto į plokštumą $\triangle ABC$ kampinių vektorių matricos $\mathbf{R}$ elementus

$$r_{1,1} = AB\cos(\angle ABC) - BC,\ r_{1,2} = AB\sin(\angle ABC),$$

$$r_{2,1} = -\cos(\angle ABC),\ r_{2,2} = -AB\sin(\angle ABC),$$

$$r_{3,1} = BC,\ r_{3,2} = 0.$$

Tegu $\triangle A^\alpha B^\alpha C^\alpha$ kampinių vektorių matricos $\mathbf{P}$ elementai lygūs

$$p_{1,1} = c_1^\alpha - b_1^\alpha,\ p_{1,2} = c_2^\alpha - b_2^\alpha,$$

$$p_{2,1} = a_1^\alpha - c_1^\alpha,\ p_{1,2} = a_2^\alpha - c_2^\alpha,$$

$$p_{3,1} = b_1^\alpha - a_1^\alpha,\ p_{3,2} = b_2^\alpha - a_2^\alpha.$$

Tada trikampių $\triangle ABC$, $\triangle A^\alpha B^\alpha C^\alpha$ 2×2 kovariacijos matrica

$$\mathbf{K} = cov(\triangle ABC, \triangle A^\alpha B^\alpha C^\alpha)$$

išreiškiama sandauga

$$\mathbf{K} = \frac{\mathbf{R}^\intercal \mathbf{Q}\mathbf{P}}{2S} = \frac{1}{2S} \begin{pmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & r_{2,2} \\ r_{3,1} & r_{3,2} \end{pmatrix}^\intercal \begin{pmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{pmatrix} \begin{pmatrix} p_{1,1} & p_{1,2} \\ p_{2,1} & p_{2,2} \\ p_{3,1} & p_{3,2} \end{pmatrix}, \tag{2.17}$$

kur S – $\triangle ABC \in T_\triangle$ plotas. Pagal 4 algoritmą išskaidžius matricą $\mathbf{K}$ singuliariosiomis reikšmėmis, sudaroma dekompozicija

$$\mathbf{K} = \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{pmatrix} \begin{pmatrix} \sigma_{1,1} & 0 \\ 0 & \sigma_{2,2} \end{pmatrix} \begin{pmatrix} v_{1,1} & v_{2,1} \\ v_{1,2} & v_{2,2} \end{pmatrix}. \tag{2.18}$$

Pagal skaitines $\sigma_{1,1}, \sigma_{2,2}$ reikšmes, apskaičiuojamas pagal (2.14), (2.15) lygybes, darbuose [27, 44] pasiūlyti trikampių panašumo ir trikampių standumo deformacijų įverčiai. Pažymėkime $t = 1, \ldots, T$ trikampių $\triangle A_t B_t C_t \in T_\triangle$ ir $\triangle A_t^\alpha B_t^\alpha C_t^\alpha \in \overline{T}_\triangle$ numerius, o atitinkamų kovariacijos matricų singuliariosios dekompozicijos elementus $\sigma_{1,1}^t$ ir $\sigma_{2,2}^t$. Tada bendras trikampių panašumo deformacijos įvertis D_{pan} išreiškiamas formule

$$D_{pan} = \frac{1}{\sum_{i=1}^T S_i} \sum_{j=1}^T S_j \Big(\frac{\sigma_{1,1}^j}{\sigma_{2,2}^j} + \frac{\sigma_{2,2}^j}{\sigma_{1,1}^j} \Big), \tag{2.19}$$

bendras trikampių deformacijos ploto įvertis D_{plot} išreiškiamas formule

$$D_{plot} = \frac{1}{\sum_{i=1}^T S_i} \sum_{j=1}^T S_j \Big(\sigma_{1,1}^j \sigma_{2,2}^j + \frac{1}{\sigma_{1,1}^j \sigma_{2,2}^j} \Big)^\Theta, \tag{2.20}$$

kur S_t – $\triangle A_t B_t C_t \in T_\triangle$ plotas, $t = 1, \ldots, T$, $\Theta \in (0, \infty)$ – parametras, skirtas standumo įverčiui normuoti.

Disertacijos 4-ame skyriuje pagal (2.19, 2.20) formules apskaičiuoti kurpalių segmentų išklotinių deformacijos įverčiai.

Kitame poskyryje apžvelgsime 3D skaitmeninių modelių saugojimo formatus, kurie buvo naudojami disertacijoje atliekant eksperimentinius tyrimus.

2.6 3D skaitmeninių modelių failų formatai

Matematiškai statinis skaitmeninis modelis apibrėžiamas grafu $G = (V, E)$, tačiau kompiuterinėje grafikoje naudojamos tekstūros, įvairūs vaizdo efektai, dinaminiai modeliai, juos formuojantys transformacijos ir deformacijos algoritmai, todėl esti įvairių formatų šiems skaitmeniniams modeliams išsaugoti.

Šioje disertacijoje skaitmeniniams modeliams sudaryti buvo naudojami OFF (angl. *object file format*) ir OBJ (angl. *wavefront object*) skaitmeninių modelių formatai. OFF formatas tiesiogiai siejamas su grafu $G = (V, E)$, tik turi privalumą – skaitmeninio modelio sienas galima nuspalvinti norimomis spalvomis. OBJ formatas apibendrina OFF formatą, nes OBJ formate pagal kontrolinius taškus galima programiškai apibrėžti parametrinius paviršius, pavyzdžiui, Bezjė kreives ar Bezjė paviršius. Yra ir daugiau skaitmeninių modelių formatų, pavyzdžiui, STL (angl. *standard tessellation language*), WRL (angl. *virtual reality modeling language*) ar BLEND (angl. *blender*). Naudojant šiuos formatus yra galimybė ant skaitmeninių modelių paviršių atvaizduoti tekstūras, naudoti specialius vaizdo efektus ar kurti dinaminius 3D skaitmeninius modelius.

Kitame poskyryje nagrinėjamos retosios matricos ir jų apdorojimo metodai, taikomi skaitmeninių modelių apdorojimui pagreitinti.

2.7 Retosios matricos ir jų apdorojimo metodai

Optimizavimo (pavyzdžiui, tiesinio programavimo) uždaviniams spręsti reikalingi specialūs metodai tikslo funkcijoms minimizuoti. Tiesinių lygčių sistemai

$$\mathbf{A}x = b \tag{2.21}$$

spręsti taikomi Gauso, Kramerio ar atvirkštinės matricos metodai reikalaujantys polinominio sudėtingumo algoritmų, tačiau šie metodai nėra efektyvūs kai matrica $\mathbf{A} = (a_{i,j})$ turi daug pasikartojančių elementų. Pavyzdžiui, darbuose [3A, 4A] įvertinus, kad $n \times n$ matricos $\mathbf{U} = (u_{i,j})$ nediagonaliniai elementai kartojasi eilutėse ir $n \times n$ matricos $\mathbf{V} = (v_{i,j})$ – stulpeliuose, ir pritaikius tiesinių lygčių sistemos

$$\begin{cases} \mathbf{U}y = \alpha\mathbf{I}, \quad 0 < \alpha \in \mathbb{R}, \quad \mathbf{I} = (1, \ldots, 1)^\intercal, \\ x\mathbf{V} = \beta\mathbf{I}, \quad 0 < \beta \in \mathbb{R}, \\ \mathbf{I}x = 1, \\ \mathbf{I}y = 1 \end{cases} \tag{2.22}$$

analitinius sprendinius $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$, darbe [66] pasiūlyto optimizavimo uždavinio sprendimo algoritmo kiekvienos iteracijos sudėtingumas pagerintas nuo polinominio iki tiesinio. Analogiškai efektyvūs matricų apdorojimo

metodai reikalingi sprendžiant paviršiaus išklotinės sudarymo optimizavimo uždavinius, nes $T_\triangle$ paviršiaus grafą apibrėžiančios matricos elementai – daugiausia nuliai.

Grafų teorijoje $|V| = n$ eilės grafą $G = (V, E)$ reprezentuoja $n \times n$ gretimumo matrica. Siekiant sudaryti tokią matricą, reikia skaičiais $1, 2, \ldots, n$ sunumeruoti grafo viršūnes. Tegu $b_{i,j}$ – skaičius briaunų, jungiančių i-ąją ir j-ąją viršūnes, $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant n$. Tokiu atveju gretimumo matrica $\mathbf{B} = (b_{i,j})$ sudaroma pagal šias taisykles:

$$b_{i,j} = \begin{cases} 1, & \text{jei viršūnes } i \text{ ir } j \text{ jungia briauna } \{i, j\}\ , \\ 0, & \text{kitu atveju.} \end{cases} \tag{2.23}$$

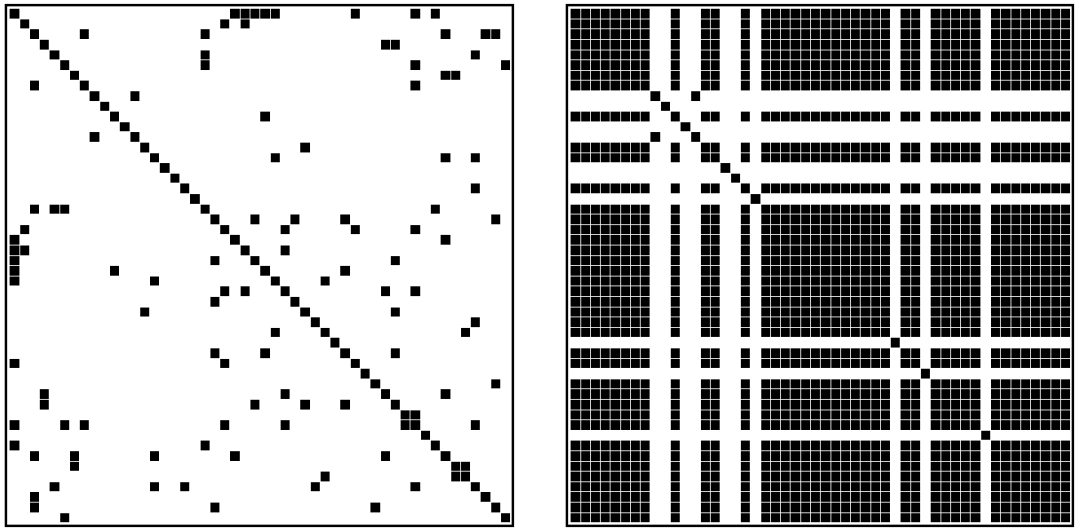
3D skaitmeninių modelių paviršiaus išklotinių sudarymo algoritmuose naudojama gretimumo matricą $\mathbf{B} = (b_{i,j})$ atitinkanti Laplaso matrica $\mathbf{L} = (l_{i,j})$:

$$l_{i,j} = \begin{cases} |Adj[i]|, & \text{jei } i = j, \\ -1, & \text{jei } i \neq j \text{ ir viršūnes } i \text{ ir } j \text{ jungia briauna } \{i, j\}, \\ 0, & \text{kitu atveju.} \end{cases} \tag{2.24}$$

Čia $|Adj[i]|$ žymi skaičių gretimų viršūnių viršūnei i. Atkreipkime dėmesį, kad nagrinėjant grafus, kurie atitinka trikampių tinklo paviršiaus skaitmeninius modelius, $|Adj[i]|$ statistiškai labiausiai tikėtina reikšmė yra 6. Tai reiškia, kad kiekvienoje matricos $\mathbf{L}$ eilutėje ir stulpelyje labiausiai tikėtina sutikti 6 reikšmes, nelygias 0. Kitaip tariant, skaitmeninius modelius atitinkančių Laplaso matricų elementai dažniausiai lygūs 0. Tokios matricos vadinamos retosiomis matricomis.

7 apibrėžimas. n-os eilės retąja matrica vadinama matrica, kurios nenulinių elementų skaičius asimptotiškai lygus $O(n)$.

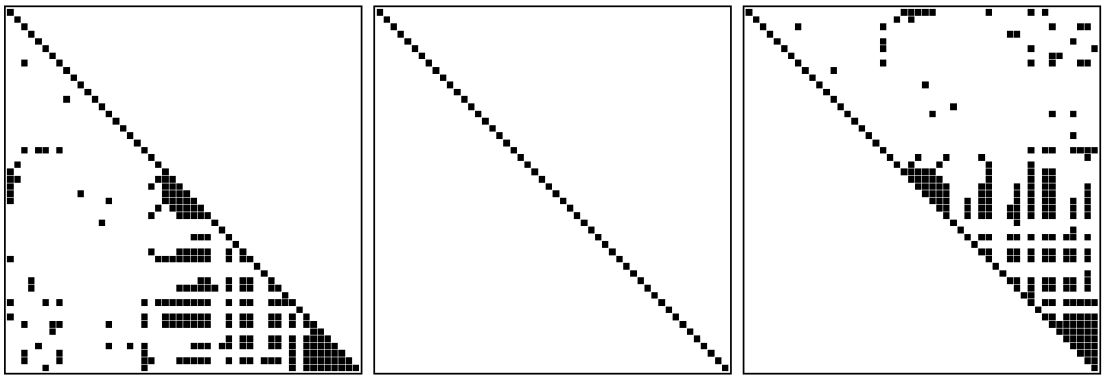
Laplaso matricų pavyzdžiai pateikti 2.8 paveiksle, kuriame juoda spalva pažymėti nenuliniai matricų elementai. Realizuojant paviršių išklotinių sudarymo algoritmus konstruojamos tikslo funkcijos, o siekiant jas minimizuoti reikia spręsti tiesinių lygčių sistemas apskaičiuojant atvirkštinę matricą pradinei retajai matricai, pavyzdžiui, Laplaso matricai $\mathbf{L}$ (2.8a paveikslas). Čia atsiranda problema, nes $\mathbf{L}^{-1}$ nėra retoji matrica (2.8b paveikslas) ir tolesniuose skaičiavimuose matricos $\mathbf{L}^{-1}$ elementai užima gana daug kompiuterio atminties, pavyzdžiui, 50 tūkstančių viršūnių turinčio skaitmeninio modelio atvirkštinė $5 \cdot 10^4 \times 5 \cdot 10^4$ Laplaso matrica



(a) Retoji Laplaso matrica $\mathbf{L}$ **(b)** Atvirkštinė Laplaso matrica $\mathbf{L^{-1}}$

2.8 paveikslas. Laplaso matricų pavyzdžiai

turėtų iki $2,5 \cdot 10^9$ skirtingų elementų. Tai užimtų iki 19 GB operatyviosios atminties naudojant 64 bitų operacinę sistemą ir $\mathbf{L}^{-1}$ apskaičiavimo užduotis taptų neišsprendžiama, jei šios operatyviosios atminties neužtektų. Todėl sprendžiant skaitmeninių modelių paviršių fragmentų išplokštinimo uždavinius naudojamos matricų faktorizacijos, pavyzdžiui, LU faktorizacija ar Choleckio dekompozicija (2.9 paveikslas).



2.9 paveikslas. Retosios Laplaso matricos Choleckio dekompozicijos pavyzdys

Apskaičiuokime $\mathbf{L}^{-1}$ taikydami Choleckio $\mathbf{MDM}^\intercal$ dekompoziciją:

$$\mathbf{L} = \mathbf{MDM}^\intercal = \begin{pmatrix} 1 & 0 & \dots & 0 \\ m_{2,1} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ m_{k,1} & m_{k,2} & \dots & 1 \end{pmatrix} \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_k \end{pmatrix} \begin{pmatrix} 1 & m_{2,1} & \dots & m_{k,1} \\ 0 & 1 & \dots & m_{k,2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix},$$

čia

$$d_j = a_{j,j} - \sum_{k=1}^{j-1} m_{j,k}^2 d_k,$$

$$m_{i,j} = \frac{1}{d_j}\left(a_{i,j} - \sum_{k=1}^{j-1} m_{i,k} m_{j,k} d_k\right), \quad \text{kai } i > j.$$

Atkreipkime dėmesį, kad $\mathbf{L}^{-1} = (\mathbf{M}^\intercal)^{-1}\mathbf{D}^{-1}\mathbf{M}^{-1}$. Tada matrica, atvirkštinė matricai $\mathbf{M}$, lygi

$$\mathbf{M}^{-1} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ c_{2,1} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_{k,1} & c_{k,2} & \dots & 1 \end{pmatrix}.$$

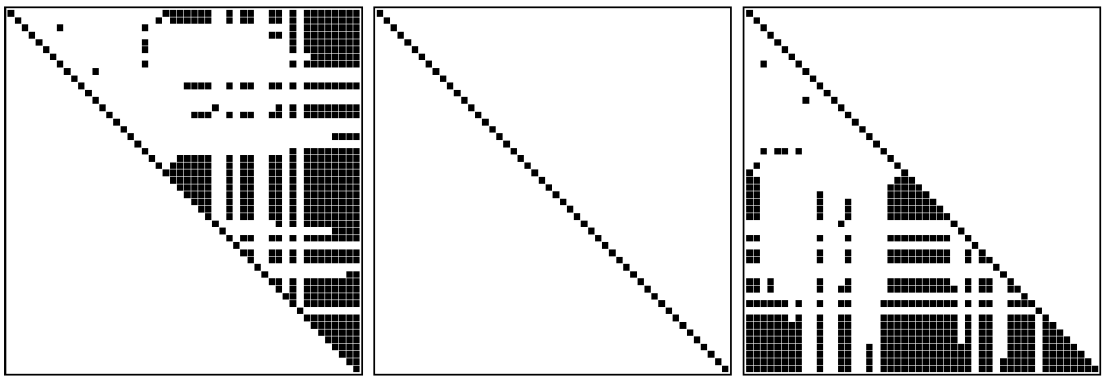
Matricos $\mathbf{M}^{-1}$ elementus $c_{i,j}$ galima apskaičiuoti pagal formules

$$c_{i,i} = 1, \quad i = 1, \dots, k,$$

$$c_{i,j} = \sum_{q=j}^{i-1} m_{i,q} c_{q,j}, \quad i = j+1, \dots, k, \quad j = 1, \dots, k.$$

Galiausiai

$$\mathbf{L}^{-1} = \begin{pmatrix} 1 & c_{2,1} & \dots & c_{k,1} \\ 0 & 1 & \dots & c_{k,2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} d_1^{-1} & 0 & \dots & 0 \\ 0 & d_2^{-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_k^{-1} \end{pmatrix} \begin{pmatrix} 1 & 0 & \dots & 0 \\ c_{2,1} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_{k,1} & c_{k,2} & \dots & 1 \end{pmatrix}.$$



2.10 paveikslas. Atvirkštinės Laplaso matricos faktorizacijos pavyzdys

Trikampių tinklo paviršių atvirkštinių Laplaso matricų faktorizacijos

$$\mathbf{L}^{-1} = (\mathbf{M}^\intercal)^{-1}\mathbf{D}^{-1}\mathbf{M}^{-1}$$

matricos $\mathbf{M}^{-1}$ ir $\mathbf{D}^{-1}$, remiantis skaitmeninių modelių gretimumo matricų statistika, užima kelis kartus mažiau operatyviosios kompiuterio atminties nei atvirkštinė Laplaso matrica $\mathbf{L}^{-1}$ (2.10 paveikslas). Šios Choleckio dekompozicijos taikymo nauda:

1. Kompiuterio atminties resursų naudojimas atvirkštinei matricai $\mathbf{L}^{-1}$ išsaugoti keičiamas skaičiavimais – atvirkštinei matricai $\mathbf{L}^{-1}$ apskaičiuoti pagal faktorizaciją $(\mathbf{M}^\intercal)^{-1}\mathbf{D}^{-1}\mathbf{M}^{-1}$.

2. Atliekami tik būtini skaičiavimai kurioje nors programavimo aplinkoje matricai $\mathbf{M}$ priskiriant retosios matricos tipą. Pavyzdžiui, C++ programavimo aplinkoje naudojant *Eigen*[8] biblioteką.

3. Galimybė lygiagretinti skaičiavimus, pavyzdžiui, naudoti *OpenMP*[9] ar kitas programavimo priemones.

2.8 Skyriaus išvados

Šiame skyriuje apžvelgti 3D skaitmeninių modelių apdorojimo algoritmai ir jų taikymas. Apžvelgti konforminiai ir izometriniai parametrizavimo metodai ir populiariausi skaitmeninių modelių paviršių išklotinių sudarymo ARAP, ABF++ ir LSCM algoritmai. Svarbiausios išvados:

1. Taikant Choleckio dekompoziciją paviršiaus išklotinių sudarymo algoritmuose kiekvienos jų tikslo funkcijos optimizavimo metu iteracijų sudėtingumas pagerinamas nuo polinominio iki tiesinio.

2. Taikant išreikštines singuliariųjų reikšmių $\sigma_{1,1}, \sigma_{2,2}$ formules (2.14), (2.15) paviršiaus deformacijos įverčiui apskaičiuoti rezultatas gaunamas kelis kartus greičiau, nes nebereikia apskaičiuoti singuliariosios dekompozicijos matricų $\mathbf{U}$ ir $\mathbf{V}$ reikšmių.

[8] Programavimo kalbos C++ algebros funkcijų, skirtų retosioms matricoms apdoroti, paketas *Eigen*. Internetinė prieiga: http://eigen.tuxfamily.org.

[9] *OpenMP* – programavimo standartas, skirtas realizuoti lygiagretiesiems algoritmams bendros atminties kompiuteriuose.

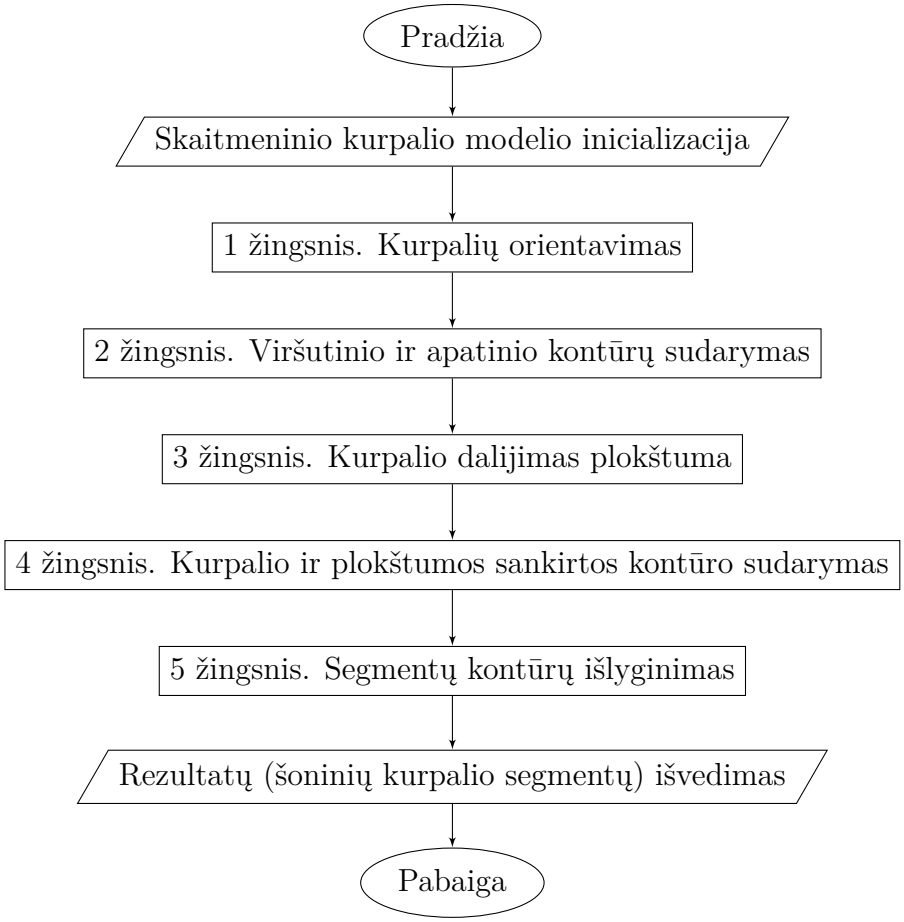
3 Nauji 3D skaitmeninių modelių apdorojimo algoritmai

Šiame skyriuje pateikiami siūlomi 3D skaitmeninių modelių apdorojimo algoritmai: kurpalių skaitmeninių modelių segmentavimo [5A, 8A] ir parametrinio C^2 paviršiaus aproksimavimo kvadrianguliariuoju paviršiumi [2A] algoritmai. Kurpalių skaitmeninių modelių segmentavimo algoritmo paskirtis – pagal ortopedijos ekspertų atliekamo rankinio kurpalių segmentavimo taisykles automatiškai kurpalių paviršių padalyti į tam tikrus skaitmeninius segmentus ir numatyti galimybę kuriuo nors paviršių išklotinių sudarymo algoritmu gauti šių segmentų išklotines. Pagal šias išklotines būtų sudaromi individualios avalynės gamybiniai brėžiniai. Parametrinio C^2 paviršiaus aproksimavimo algoritmas gali būti taikomas iš taškų debesų rankiniu būdu modeliuojant parametrinį paviršių, pavyzdžiui, parametrinį kurpalio paviršių, kuris norimu tikslumu būtų aproksimuotas kvadrianguliariuoju paviršiumi. Taikant šį algoritmą būtų sudaromas aukštos kokybės paviršius, aprašomas matematinėmis formulėmis, ir tausojami kompiuterio atminties resursai, nes atsirastų galimybė norimu tikslumu šį tolyduįį (parametrinį) matematinį paviršių atvaizduoti į diskretųjį paviršių (skaitmeninį modelį).

3.1 Naujas kurpalių skaitmeninių modelių segmentavimo algoritmas

Skaitmeninių modelių segmentavimas taikomas siekiant klasterizuoti trimačio objekto paviršiaus sritis pagal spalvą, formą, viršūnių tankumą ar kitus kriterijus. Dažniausiai praktikoje naudojamas segmentavimas siekiant atskirti tokias paviršiaus sritis, ant kurių atvaizduotos tekstūros trimatėje erdvėje būtų kuo mažiau deformuotos plokštumoje. Tokio segmentavimo nauda – tikslesnės trimačio objekto paviršiaus fragmentų išklotinės, aukštesnės vaizdo kokybės paveikslėliai

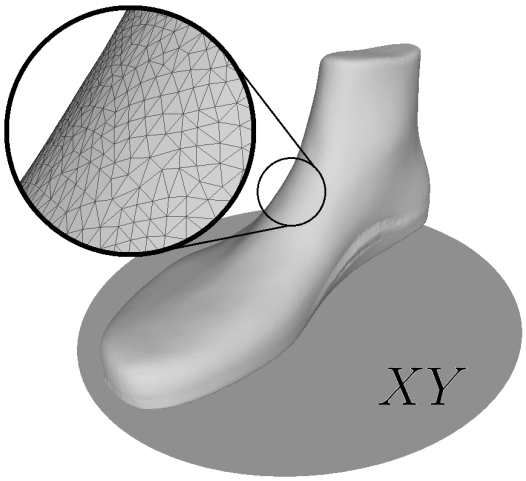
plokštumoje ir mažiau kompiuterio atminties užimančios tekstūros. Šie algoritmai grindžiami skaitmeninio modelio „karpymu“ pagal paviršiaus taško grandiento kryptį arba paviršiaus klasterizavimu pagal normalių kryptis. Šiame skyriuje pristatysime segmentavimo algoritmą, skirtą kurpalių 3D skaitmeninių modelių paviršiui padalyti į sritis, kurių išklotinių kontūrai atitiktų vienetinių kurpalių 2D gamybinius brėžinius. Pagrindiniai algoritmo žingsniai pavaizduoti 3.1 paveiksle.



3.1 paveikslas. Kurpalio paviršiaus segmentavimo algoritmo blokinė schema

3.1.1 1 žingsnis. Kurpalių orientavimas

Skaitmeninami kurpaliai pastatomi vertikaliai ant skenerio, t. y. padu į apačią, todėl skaitmeniniai jų modeliai atitinkamai padėti ant XY plokštumos (3.2 paveikslas) stačiakampėje Dekarto koordinačių sistemoje. Taigi laikysime, kad tai pradinė būtinoji sąlyga. Jei kurpalis kitaip orientuotas, jį būtina pastatyti ant XY plokštumos.



3.2 paveikslas. Kurpalio, padėto ant XY plokštumos, 3D skaitmeninis modelis

3.1.2 2 žingsnis. Viršutinio ir apatinio kontūrų sudarymas

Kadangi kurpalis padėtas ant XY plokštumos, jo pado trikampių normalių kryptys nukreiptos žemyn, t. y. į XY plokštumą, o kampai, kuriuos sudaro normalių kryptys su šia plokštuma, priklauso intervalui $[-\frac{\pi}{2}; -\frac{\pi}{4}]$. Tegu $\triangle ABC$ – bet kuris skaitmeninio kurpalio modelio trikampis, kurio viršūnės išdėstytos pagal laikrodžio kryptį iš skaitmeninio modelio išorės pusės. Tegu $\overrightarrow{a}(a_1, a_2, a_3) = \overrightarrow{BC}$, $\overrightarrow{b}(b_1, b_2, b_3) = \overrightarrow{CA}$, tada $\triangle ABC$ normalė lygi

$$\overrightarrow{n}_{\triangle ABC}(n_1, n_2, n_3) = \overrightarrow{n}_{\triangle ABC}(a_2b_3 - a_3b_2, a_3b_1 - a_1b_3, a_1b_2 - a_2b_1). \tag{3.1}$$

Kampas tarp normalės $\overrightarrow{n}_{\triangle ABC}(n_1, n_2, n_3)$ ir XY plokštumos normalės $\overrightarrow{n}_{XY}(0, 0, 1)$ lygus

$$\alpha_{\triangle ABC} = \arcsin\left(\frac{n_3}{\sqrt{n_1^2 + n_2^2 + n_3^2}}\right). \tag{3.2}$$

Tegu $G = (V, E)$ yra pradinis grafas, atitinkantis trianguliuotąjį skaitmeninio kurpalio modelį. Pažymėkime šio grafo viršūnių poaibius $V_1, V_2, V_3 \subset V$:

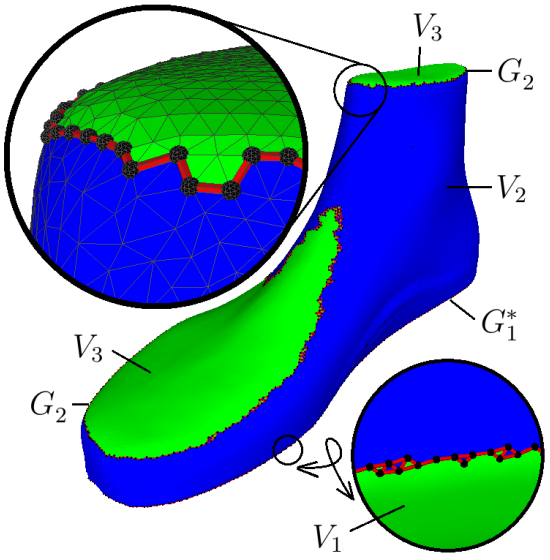
$$V_1 = \big\{v \subset V : A, B, C \in v, -\pi/2 \leqslant \alpha_{\triangle ABC} \leqslant -\pi/4\big\}, \tag{3.3}$$

$$V_2 = \big\{v \subset V : A, B, C \in v, -\pi/4 < \alpha_{\triangle ABC} < \pi/4\big\}, \tag{3.4}$$

$$V_3 = \big\{v \subset V : A, B, C \in v, \pi/4 \leqslant \alpha_{\triangle ABC} \leqslant \pi/2\big\}. \tag{3.5}$$

Iš grafų (3.3 paveikslas)

$$G_1 = \Big(V_1 \cap V_2, \big\{\{u, v\} \in E : u, v \in V_1 \cap V_2\big\}\Big), \tag{3.6}$$



3.3 paveikslas. Kurpalio 3D skaitmeninio modelio segmentų, sudarytų pagal normalių kryptis, sankirta.

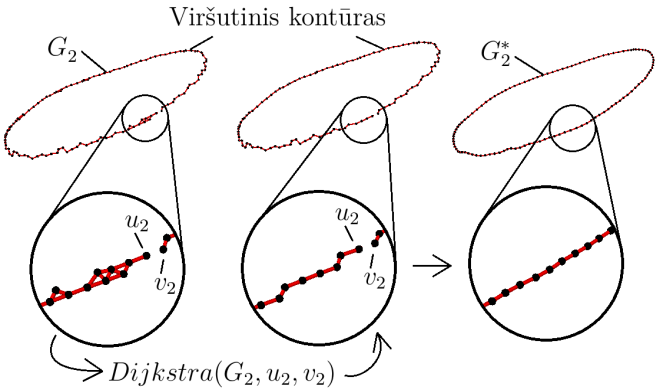
$$G_2 = \Big(V_2 \cap V_3, \big\{\{u,v\} \in E : u,v \in V_2 \cap V_3\big\}\Big) \tag{3.7}$$

atitinkamai pašalinę briaunas $\{u_1, v_1\}$, $\{u_2, v_2\}$, kad $|Adj[u_1]| = 2$ ir $|Adj[u_2]| = 2$, siekdami, kad egzistuotų vieninteliai trumpiausi takai $u_1 \rightsquigarrow v_1$, $u_2 \rightsquigarrow v_2$, ir realizavę Dijkstros trumpiausių takų algoritmus tarp viršūnių $u_1, v_1 \in V_1 \cap V_2$ ir $u_2, v_2 \in V_2 \cap V_3$, gauname 2 naujus grafų takus $u_1 \rightsquigarrow v_1$, $u_2 \rightsquigarrow v_2$. Prie jų atitinkamai pridėję išmestas briaunas $\{u_1, v_1\}$, $\{u_2, v_2\}$ galiausiai gauname ciklinius grafus

$$G_1^* = \Big(V_1^*, \big\{\{u,v\} \in u_1 \rightsquigarrow v_1 \cup \{u_1, v_1\}\big\}\Big), \tag{3.8}$$

$$G_2^* = \Big(V_2^*, \big\{\{u,v\} \in u_2 \rightsquigarrow v_2 \cup \{u_2, v_2\}\big\}\Big), \tag{3.9}$$

atitinkančius viršutinį ir apatinį kurpalių kontūrus (3.4 paveikslas).



3.4 paveikslas. Kurpalio kontūrų apdorojimas

3.1.3 3 žingsnis. Kurpalio dalijimas plokštuma

Plokštumą erdvėje apibrėžia 3 taškai, nesantys toje pačioje tiesėje. Kurpalį dalijant plokštuma išilgai pagrindinė užduotis – gauti 2 segmentus (kurpalio šonus), kurie atvaizduojami į plokštumą būtų kaip galima mažiau deformuojami (išlaikomi trianguliariojo paviršiaus trikampių kampai ar kraštinių ilgiai). Remiantis heuristika, pagal ortopedijos ekspertų atliekamą kurpalių paviršių dalijimą nustatyta, kad pirmasis iš trijų plokštumos taškų atitinka viršutinio G_1^* kontūro taškų aritmetinį vidurkį, kiti 2 taškai priklauso tiesei, kuri eina išilgai per apatinį kontūrą ir geriausiai jį tiesiškai koreliuoja šiam kontūrui priklausančių viršūnių atžvilgiu. Disertacijoje siūlomas algoritmas, kuriuo randami tokie 2 taškai grindžiamas apatinio kontūro segmentų sudarymu ir šiems segmentams priklausančių taškų aritmetinių vidurkių apskaičiavimu. Tegu $A(x_A, y_A, z_A)$ ir $B(x_B, y_B, z_B)$ – du vienas nuo kito labiausiai nutolę apatiniam kontūrui priklausantys taškai ir $C(x_C, y_C, z_C)$ – bet kuris apatinio kontūro taškas. Tegu $t \in [0, 1]$ – parametras, parodantis, kurioje vietoje atkarpą AB kirstų plokštuma $\gamma(t)$

$$\gamma(t) : \sum_{k\in\{x,y,z\}} \Big(k - k_A(1-t) - k_B t\Big)\Big(k_A - k_B\Big) = 0,$$

ortogonali pradinei atkarpai AB. Tegu

$$\delta(t, C) = \sum_{k\in\{x,y,z\}} \Big(k_C - k_A(1-t) - k_B t\Big)\Big(k_A - k_B\Big). \tag{3.10}$$

Pažymėkime apatinio kontūro G_1^* viršūnių aibes:

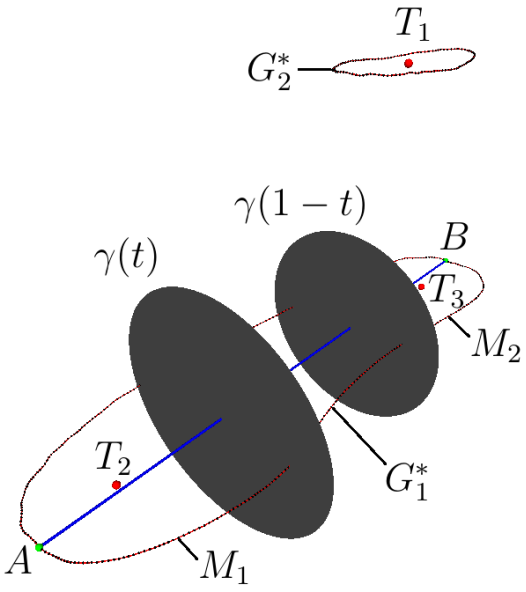
$$M_1 = \big\{v \in V_1^* : \delta(t, v) > 0\big\}, \tag{3.11}$$

$$M_2 = \big\{v \in V_1^* : \delta(1-t, v) < 0\big\}. \tag{3.12}$$

Remiantis atliktų eksperimentų heuristika, nustatyta, kad atkarpą AB geriausia dalyti į 3 dalis, tada $t = \frac{1}{3}$.

M_1 aibė priklauso puserdvei, ribojamai plokštumos $\gamma(t)$, arčiau taško A, M_2 aibė priklauso puserdvei, ribojamai plokštumos $\gamma(1-t)$, arčiau taško B (3.5 paveikslas). Taigi kurpalį išilgai dalija plokštuma, einanti per taškus $T_j(x_{T,j}, y_{T,j}, z_{T,j}), j = 1, 2, 3$:

$$T_1 = \frac{1}{|V_1^*|}\sum_{v\in V_1^*} v, \quad T_2 = \frac{1}{|M_1|}\sum_{v\in M_1} v, \quad T_3 = \frac{1}{|M_2|}\sum_{v\in M_2} v. \tag{3.13}$$



3.5 paveikslas. Kurpalį dalijančios plokštumos apskaičiavimas

Belieka suprojektuoti kontūrą G_3^* į plokštumą α:

$$\alpha : \begin{vmatrix} x - x_{T,1} & y - y_{T,1} & z - z_{T,1} \\ x_{T,2} - x_{T,1} & y_{T,2} - y_{T,1} & z_{T,2} - z_{T,1} \\ x_{T,3} - x_{T,2} & y_{T,3} - y_{T,2} & z_{T,3} - z_{T,2} \end{vmatrix} = ax + by + cz + d = 0,$$

iš čia

$$a = y_{T,1}z_{T,2} - y_{T,1}z_{T,3} - z_{T,1}y_{T,2} + z_{T,1}y_{T,3} + y_{T,2}z_{T,3} - z_{T,2}y_{T,3}, \tag{3.14}$$

$$b = -x_{T,1}z_{T,2} + x_{T,1}z_{T,3} + z_{T,1}x_{T,2} - z_{T,1}x_{T,3} - x_{T,2}z_{T,3} + z_{T,2}x_{T,3}, \tag{3.15}$$

$$c = x_{T,1}y_{T,2} - x_{T,1}y_{T,3} - y_{T,1}x_{T,2} + y_{T,1}x_{T,3} + x_{T,2}y_{T,3} - y_{T,2}x_{T,3}, \tag{3.16}$$

$$d = -x_{T,1}a - y_{T,1}b - z_{T,1}c. \tag{3.17}$$

3.1.4 4 žingsnis. Kurpalio ir plokštumos sankirtos kontūro sudarymas

Pirmoji viršūnė v_0^*, priklausanti kontūrui $G_3^*(V_3^*, E_3^*)$, priskiriama viršūnei $v \in V$, kuri yra arčiausiai plokštumos $\gamma(t)$, t. y. pagal $\min_{v \in V} d(\alpha, v)$. Čia $d(\alpha, v)$ – atstumas nuo viršūnės $v(x_v, y_v, z_v)$ iki plokštumos α. Šios viršūnės projekcija $h(v, \alpha)$ plokštumoje α lygi

$$h(v, \alpha) = v^\alpha(x_v + ta, y_v + tb, z_v + tc), \quad \text{kur} \quad t = -\frac{ax_v + by_v + cz_v + d}{a^2 + b^2 + c^2}, \tag{3.18}$$

čia a, b, c, d – plokštumos α parametrai. Toliau randame artimiausią plokštumai α viršūnę v_1^*, kuri gretima viršūnei v_0^*, ir ją suprojektuojame į plokštumą α pagal (3.18). Grafą G_3^* sudarome iš v_0^* ir v_1^* viršūnių: $G_3^* = (\{v_0^*, v_1^*\}, \{\{v_0^*, v_1^*\}\})$. Kitas viršūnes $v_2^*, v_3^*, v_4^*, \ldots$ į grafą G_3^* įterpiame pagal 5 algoritmą.

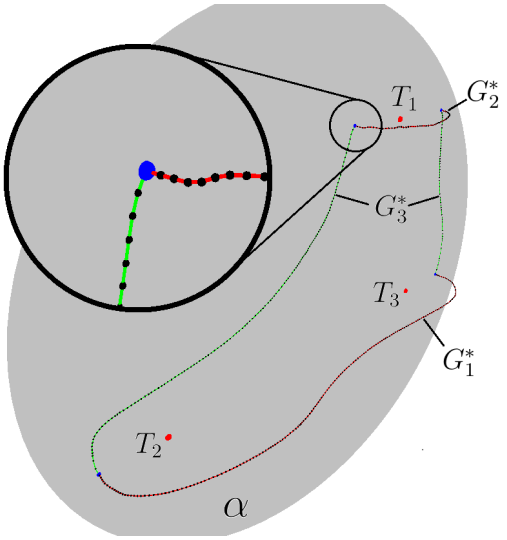
5 algoritmas. Kurpalio skaitmeninio modelio ir jį pusiau dalijančios plokštumos sankirtos kontūro sudarymas

Įvesties duomenys : Grafas G, 2 pradinės kontūro G_3^* viršūnės v_0^*, v_1^*

Išvesties duomenys: Kontūras G_3^*

1 $G_3^* \leftarrow (\{v_0^*, v_1^*\}, \{v_0^* v_1^*\})$;
2 $i \leftarrow 1$;
3 $v_{i+1}^* \leftarrow v_1^*$;
4 **while** $v_{i+1}^* \neq v_0^*$ **do**
5 $\overrightarrow{m_1} \leftarrow \overrightarrow{v_{i-1}^* v_i^*}$;
6 $v_{i+1}^* \leftarrow Adj[v_i^*][1]$;
7 **for *each*** $v \in Adj[v_i^*]$ **do**
8 $\overrightarrow{m_2} \leftarrow \overrightarrow{v_i^* v}$;
9 **if** $\angle(\overrightarrow{m_1}, \overrightarrow{m_2}) < \pi/2$ **then**
10 **if** $d(v_{i+1}^*, \alpha) > d(v, \alpha)$ **then**
11 $v_{i+1}^* \leftarrow v$
12 **end**
13 **end**
14 **end**
15 $v_{i+1}^* \leftarrow h(v_{i+1}^*, \alpha)$ pagal (3.18);
16 $G_3^* \leftarrow G_3^* \cup (\{v_{i+1}^*\}, \{v_i^* v_{i+1}^*\})$;
17 $i \leftarrow i + 1$;
18 **end**

Šio algoritmo idėja – sudaryti kontūrą G_3^* pridedant vis po vieną viršūnę v_{i+1}^* ir išlaikant buvusią viršūnių dėjimo kryptį (vektorius $\overrightarrow{m_1}$). Kai v_{i+1}^* ir v_0^* sutampa, algoritmas nebekartojamas, o suformuojamas uždaras kontūras G_3^*. 5 algoritmo eigoje sudarant kontūrą G_3^* kertami G_1^* ir G_2^* kontūrai, tačiau, algoritme po šių kontūrų kirtimosi pridėjus papildomas viršūnių v_{i+1}^* įtraukimo į kontūrą G_3^* sąlygas, galima kontūrą G_3^* išskaidyti į 2 dalis (3.6 paveikslas), nors tai ir neturi įtakos įvesties grafo G šoninių segmentų sudarymui.



3.6 paveikslas. Kurpalio dalijimas plokštuma

3.1.5 5 žingsnis. Segmentų sudarymas ir kontūrų išlyginimas

Siekiant kontūrams $G_1^* = (V_1^*, E_1^*), G_2^* = (V_2^*, E_2^*), G_3^* = (V_3^*, E_3^*)$ priklausančias viršūnes išdėstyti tolygiai, taikoma slenkančiojo vidurkio formulė – viršūnių $v^*(x_v^*, y_v^*, z_v^*) \in \{V_1^*, V_2^*, V_3^*\}$ koordinatės keičiamos naujomis ir gaunama kitų viršūnių aibė $\hat{v}(\hat{x}_v^*, \hat{y}_v^*, \hat{z}_v^*) \in \{\hat{V}_1^*, \hat{V}_2^*, \hat{V}_3^*\}$ taikant formulę

$$\hat{v}^* = \frac{1}{2}v^* + \frac{1}{4}\sum_{u \in Adj[v^*]} u. \tag{3.19}$$

Priskyrus V_1^*, V_2^*, V_3^* atitinkamai $\hat{V}_1^*, \hat{V}_2^*, \hat{V}_3^*$ sudaromi lygesni kontūrai G_1^*, G_2^*, G_3^*. Šį algoritmą galima kartoti 2–3 kartus, vis labiau lyginant kontūrus.

Galiausiai lieka tik išvesti rezultatus, t. y. skaitmeninio kurpalio modelio šoninius segmentus $\widetilde{G}_1, \widetilde{G}_2$. Kontūrai G_1^*, G_2^*, G_3^* riboja kairę ir dešinę kurpalio pusę atitinkančius lekalus. Užtenka rasti viršūnes $\widetilde{v}_1 \in \widetilde{G}_1, \widetilde{v}_2 \in \widetilde{G}_2$ skirtingose kurpalio pusėse, tada realizuoti paieškos į plotį algoritmus $BFS(G, \widetilde{v}_1), BFS(G, \widetilde{v}_2)$ [55] su kontūrų G_1^*, G_2^*, G_3^* apribojimais (3.7 paveikslas) ir atrastas viršūnes priskirti šoniniams kurpalio segmentams $\widetilde{G}_1, \widetilde{G}_2$. Vienas iš būtų parinkti pradines viršūnes $\widetilde{v}_1, \widetilde{v}_2$ paieškai į plotį grafe G – rasti į skirtingas puses labiausiai nutolusias viršnes nuo plokštumos α:

$$\widetilde{v}_1(\widetilde{x}_{v,1}, \widetilde{y}_{v,1}, \widetilde{z}_{v,1}) \leftarrow v \in V : \max(a\widetilde{x}_{v,1} + b\widetilde{y}_{v,1} + c\widetilde{z}_{v,1} + d),$$

$$\widetilde{v}_2(\widetilde{x}_{v,2}, \widetilde{y}_{v,2}, \widetilde{z}_{v,2}) \leftarrow v \in V : \min(a\widetilde{x}_{v,2} + b\widetilde{y}_{v,2} + c\widetilde{z}_{v,2} + d).$$

6 algoritmas. Kurpalio 3D modelio šoninių segmentų sudarymas

Įvesties duomenys : Grafas $G = (V, E)$

Išvesties duomenys: Grafo $G = (V, E)$ šoniniai segmentai (pografiai)

$$\widetilde{G}_1 = (\widetilde{V}_1, \widetilde{E}_1), \widetilde{G}_2 = (\widetilde{V}_2, \widetilde{E}_2)$$

for ***each*** $\triangle ABC \in G$ **do**

  $\overrightarrow{BC} \leftarrow \overrightarrow{a}(a_1, a_2, a_3), \overrightarrow{CA} \leftarrow \overrightarrow{b}(b_1, b_2, b_3)$;

  $\alpha_{\triangle ABC} \leftarrow \arcsin\left(\frac{a_1b_2 - a_2b_1}{\sqrt{(a_2b_3 - a_3b_2)^2 + (a_3b_1 - a_1b_3)^2 + (a_1b_2 - a_2b_1)^2}}\right)$;

end

$V_1 \leftarrow \big\{v \subset V : A, B, C \in v, -\pi/2 \leqslant \alpha_{\triangle ABC} \leqslant -\pi/4\big\}$;

$V_2 \leftarrow \big\{v \subset V : A, B, C \in v, -\pi/4 < \alpha_{\triangle ABC} < \pi/4\big\}$;

$V_3 \leftarrow \big\{v \subset V : A, B, C \in v, \pi/4 \leqslant \alpha_{\triangle ABC} \leqslant \pi/2\big\}$;

$G_1 \leftarrow \Big(V_1 \cap V_2, \big\{\{u, v\} \in E : u, v \in V_1 \cap V_2\big\}\Big)$;

$G_2 \leftarrow \Big(V_2 \cap V_3, \big\{\{u, v\} \in E : u, v \in V_2 \cap V_3\big\}\Big)$;

$u_1 \rightsquigarrow v_1 \leftarrow Dijkstra\big(G_1\big(V_1, E_1 \setminus \{u_1v_1\}\big), u_1, v_1\big) : |Adj[u_1]| = |Adj[v_1]| = 2$;

$u_2 \rightsquigarrow v_2 \leftarrow Dijkstra\big(G_2\big(V_2, E_2 \setminus \{u_2v_2\}\big), u_2, v_2\big) : |Adj[u_2]| = |Adj[v_2]| = 2$;

$G_1^* \leftarrow \Big(V_1^*, \big\{\{u, v\} \in u_1 \rightsquigarrow v_1 \cup u_1v_1\big\}\Big)$;

$G_2^* \leftarrow \Big(V_2^*, \big\{\{u, v\} \in u_2 \rightsquigarrow v_2 \cup u_2v_2\big\}\Big)$;

$V_2^* \ni A, B \leftarrow G_2^* : \max(d(A, B))$;

$M_1 \leftarrow \big\{v \in V_1^* : \delta(t, v) > 0\big\},\ M_2 \leftarrow \big\{v \in V_1^* : \delta(1 - t, v) < 0\big\}$ pagal (3.10);

$T_j(x_{T,j}, y_{T,j}, z_{T,j}) : T_1 \leftarrow \frac{1}{|V_1^*|} \sum_{v \in V_1^*} v,\ T_2 \leftarrow \frac{1}{|M_1|} \sum_{v \in M_1} v,\ T_3 \leftarrow \frac{1}{|M_2|} \sum_{v \in M_2} v$;

$a \leftarrow y_{T,1}z_{T,2} - y_{T,1}z_{T,3} - z_{T,1}y_{T,2} + z_{T,1}y_{T,3} + y_{T,2}z_{T,3} - z_{T,2}y_{T,3}$;

$b \leftarrow -x_{T,1}z_{T,2} + x_{T,1}z_{T,3} + z_{T,1}x_{T,2} - z_{T,1}x_{T,3} - x_{T,2}z_{T,3} + z_{T,2}x_{T,3}$;

$c \leftarrow x_{T,1}y_{T,2} - x_{T,1}y_{T,3} - y_{T,1}x_{T,2} + y_{T,1}x_{T,3} + x_{T,2}y_{T,3} - y_{T,2}x_{T,3}$;

$d \leftarrow -x_{T,1}a - y_{T,1}b - z_{T,1}c,\ \alpha : ax + by + cz + d = 0$;

$v_0^* \leftarrow v \in V : \min(d(\alpha, v))$;

$v_1^* \leftarrow v \in Adj[v_0^*] : \min(d(\alpha, v))$;

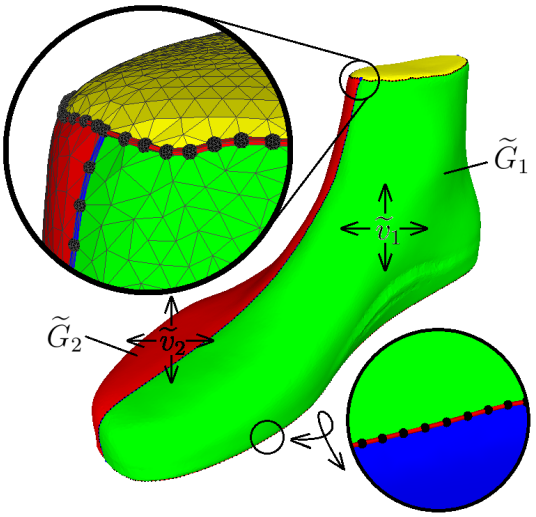
$G_3^* \leftarrow 5$ algoritmas;

$V_1^* \leftarrow \hat{V}_1^* \leftarrow V_1^*, V_2^* \leftarrow \hat{V}_2^* \leftarrow V_2^*, V_3^* \leftarrow \hat{V}_3^* \leftarrow V_3^*$ pagal (3.19);

$\widetilde{v}_1(\widetilde{x}_{v,1}, \widetilde{y}_{v,1}, \widetilde{z}_{v,1}) \leftarrow v \in V : \max(a\widetilde{x}_{v,1} + b\widetilde{y}_{v,1} + c\widetilde{z}_{v,1} + d)$;

$\widetilde{v}_2(\widetilde{x}_{v,2}, \widetilde{y}_{v,2}, \widetilde{z}_{v,2}) \leftarrow v \in V : \min(a\widetilde{x}_{v,2} + b\widetilde{y}_{v,2} + c\widetilde{z}_{v,2} + d)$;

$\widetilde{G}_1 \leftarrow BFS(G, \widetilde{v}_1), \widetilde{G}_2 \leftarrow BFS(G, \widetilde{v}_2)$ su G_1^*, G_2^*, G_3^* apribojimais;



3.7 paveikslas. Galutinių kurpalio segmentų sudarymas

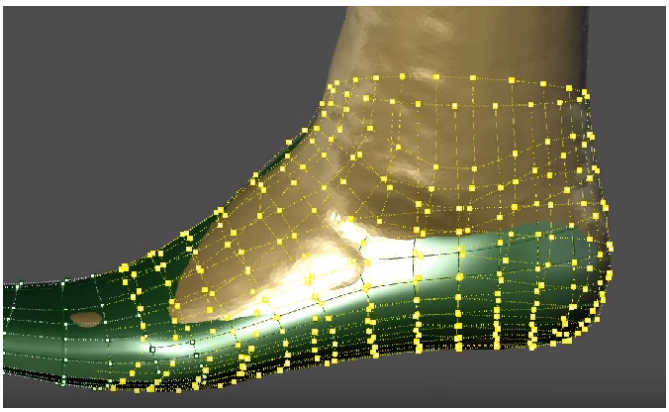
6 algoritmas parodo pagrindinius kurpalio šoninių segmentų sudarymo žingsnius. Disertacijos prieduose pateiktas skaitmeninio kurpalio modelio paviršiaus segmentavimo algoritmo pirminis programos tekstas, skirtas *Maple* programavimo aplinkai.

3.2 Parametrinio paviršiaus konstravimas iš trimačio taškų debesies

Šiame poskyryje nagrinėsime disertacijoje siūlomą metodą, kuris yra gerai žinomo C^2 paviršiaus konstravimo iš atskirų trečios eilės Bezjė paviršių algoritmo [47] patobulinimas.

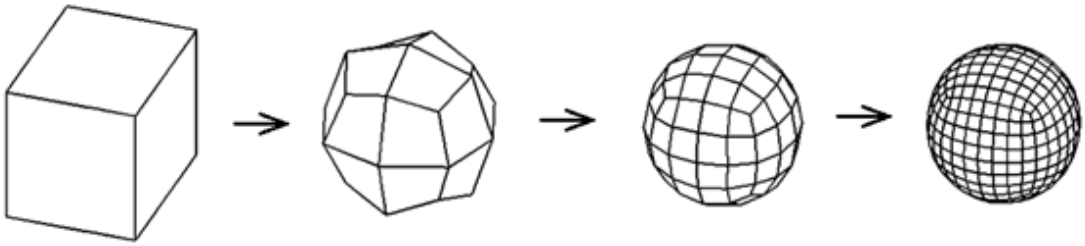
Pirminiai skaitmeniniai kurpalių modeliai daromi nuskenuotų žmogaus pėdų taškų debesis rankiniu būdu aproksimuojant NURBS paviršiais. Pasitelkiamos CAD programos ir keičiant kontrolinių taškų koordinates sumodeliuojamas NURBS paviršius [82], apgaubiantis pėdos taškų debesį ir atitinkantis būsimo bato dizainą (3.8 paveikslas). Parametrinio NURBS paviršiaus aproksimaciją – keturkampių tinklą visada galima konvertuoti į trikampių tinklą kiekvieną keturkampį padalijant į du trikampius. Tai leidžia sudaryti keturkampio tinklo paviršiaus išklotines naudojant disertacijoje apžvelgtus paviršių išklotinių sudarymo algoritmus.

Siekiant sudaryti aukštos kokybės parametrinį paviršių naudojamas NURBS



3.8 paveikslas. Skaitmeninio kurpalio modelio sudarymas pagal taškų debesį CAD aplinkoje

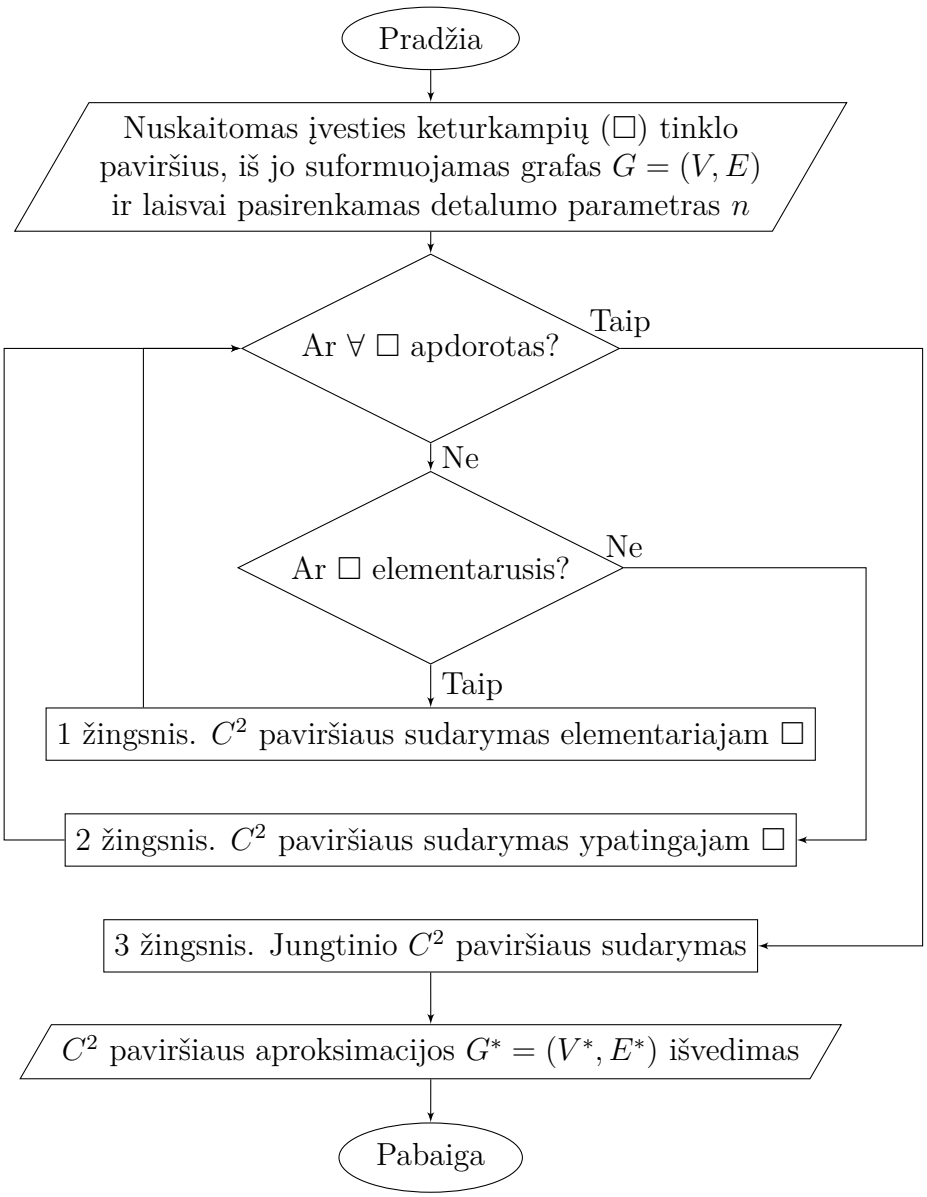
poklasis – C^2 paviršių klasė. Ši paviršių klasė ypatinga tuo, kad parametrinių paviršių sandūroje sutampa antros eilės išvestinės (paviršiaus kreivis) kiekvieno parametrinio paviršiaus atžvilgiu. Vienas iš būdų sudaryti C^2 paviršių – trečios eilės Bezjė paviršius sujungti tarpusavyje. Šis algoritmas labai patogus, kai siekiama norimu tikslumu aproksimuoti parametrinį C^2 paviršių [46]. Viena esminių problemų – C^2 paviršiaus sudarymas ypatingųjų viršūnių aplinkoje. Ypatingąja viršūne vadinama vidinė viršūnė, turinti ne 4 gretimas viršūnes. Pagal [47] algoritmą sudarant C^2 paviršių duotam keturkampių tinklui, keturkampiai, turintys bent vieną ypatingąją viršūnę, dalijami į 4 naujus keturkampius, taikant Catmullo ir Clarko [20] paviršių dalijimo algoritmą (3.9 paveikslas). Sudarytiems naujiems



3.9 paveikslas. Catmullo ir Clarko algoritmo trijų iteracijų vizualizacija.

keturkampiams apskaičiuojamos Bezjė paviršių lygtys.

Akivaizdu, kad šis dalijimo procesas begalinis, todėl ypatingųjų viršūnių aplinkose gaunamas naujų viršūnių sutankėjimas. Disertacijoje siūloma šį sutankėjimą likviduoti apskaičiuojant tik reikiamų viršūnių koordinates ir tokiu būdu begalinį paviršių dalijimo procesą keisti baigtiniu. Siūlomo algoritmo blokinė pateikta 3.10 paveiksle.



3.10 paveikslas. C^2 paviršiaus aproksimacinio keturkampių tinklo sudarymas

Elementariuoju keturkampiu vadinsime keturkampį, kurio visos viršūnės nėra ypatingosios; ypatinguoju keturkampiu vadinsime keturkampį, kurio bent viena viršūnė – ypatingoji.

3.2.1 1 žingsnis. C^2 paviršiaus sudarymas elementariajam keturkampiui

Šiame poskyryje nagrinėjamas standartinis C^2 paviršiaus sudarymo iš trečios eilės Bezjė paviršių algoritmas.

3.2.1.1 Trečios eilės Bezjė paviršius

Tegu $B(u,v)$ – trečios eilės Bezjė paviršius, kurio parametrinė forma

$$B(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} \binom{3}{i} u^i (1-u)^{n-i} \binom{3}{j} v^j (1-v)^{m-j} P_{i,j}, \tag{3.20}$$

kur $u, v \in [0,1]$ – paviršiaus parametrai, $P_{i,j}$ – kontroliniai taškai,

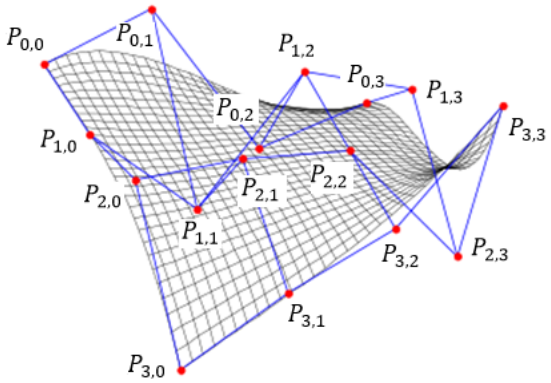
$\binom{n}{k} = \frac{n!}{k!(n-k)!}$ – derinių skaičius.

Apskaičiavus binominius koeficientus, paviršiaus parametrinę lygtį galima suvesti į matricinę formą:

$$B(u,v) = \begin{pmatrix} 1 & u & u^2 & u^3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} \begin{pmatrix} P_{0,0} & P_{0,1} & P_{0,2} & P_{0,3} \\ P_{1,0} & P_{1,1} & P_{1,2} & P_{1,3} \\ P_{2,0} & P_{2,1} & P_{2,2} & P_{2,3} \\ P_{3,0} & P_{3,1} & P_{3,2} & P_{3,3} \end{pmatrix} \cdot$$

$$\cdot \begin{pmatrix} 1 & -3 & 3 & -1 \\ -3 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ v \\ v^2 \\ v^3 \end{pmatrix}.$$

$P_{i,j}$ – kontroliniai taškai, pagal kurių išsidėstymą suformuojamas Bezjė paviršius (3.11 paveikslas) arba kitaip – C^2 paviršiaus fragmentas.

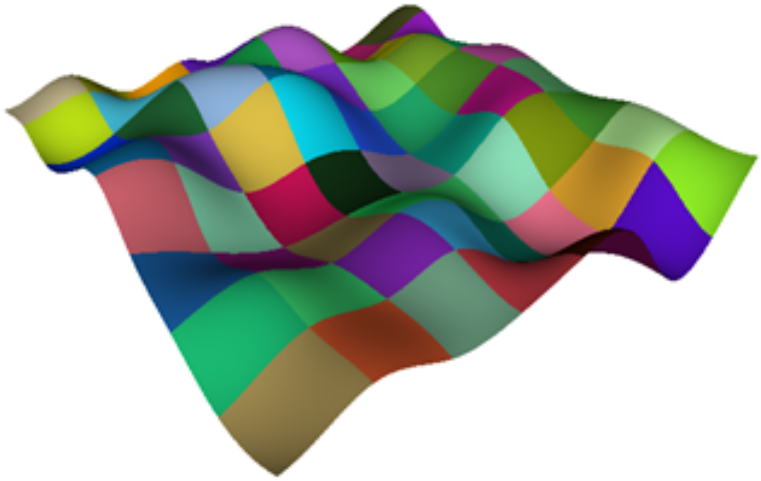


3.11 paveikslas. Bezjė paviršiaus pavyzdys

3.2.1.2 Trečios eilės Bezjė paviršių jungimas į C^2 paviršių

C^2 paviršių galima sudaryti iš 3 eilės Bezjė paviršių, kurių sandūroje sutaptų antros eilės išvestinės bet kokio tiesaus pjūvio plokštumos atžvilgiu. 3.12 paveiksle pateiktas C^2 paviršiaus pavyzdys, kuriame skirtingos spalvos paviršius

atitinka skirtingą Bezjė paviršių. Pagrindinis uždavinys – apskaičiuoti šių Bezjė paviršių kontrolinius taškus. Tarkime, turime kvadrianguliaryjį paviršių, kuriame



3.12 paveikslas. C^2 paviršius, sudarytas iš Bezjė paviršių

iš kiekvienos vidinės viršūnės išeina po 4 briaunas. Tada kiekvienam vidiniam šio paviršiaus keturkampiui galima sukonstruoti Bezjė paviršių, kurio kontroliniai taškai apskaičiuojami pagal šias formules [70]:

$$P_{0,0} = \frac{1}{36}A_{0,0} + \frac{1}{36}A_{0,2} + \frac{1}{36}A_{2,0} + \frac{1}{36}A_{2,2} + \frac{1}{9}A_{0,1} + \frac{1}{9}A_{1,0} + \frac{1}{9}A_{1,2} + \frac{1}{9}A_{2,1} + \frac{4}{9}A_{1,1},$$

$$P_{0,1} = \frac{1}{18}A_{0,2} + \frac{1}{18}A_{2,2} + \frac{1}{9}A_{0,1} + \frac{1}{9}A_{2,1} + \frac{2}{9}A_{1,2} + \frac{4}{9}A_{1,1},$$

$$P_{0,2} = \frac{1}{18}A_{0,1} + \frac{1}{18}A_{2,1} + \frac{1}{9}A_{0,2} + \frac{1}{9}A_{2,2} + \frac{2}{9}A_{1,1} + \frac{4}{9}A_{1,2},$$

$$P_{0,3} = \frac{1}{36}A_{0,1} + \frac{1}{36}A_{0,3} + \frac{1}{36}A_{2,1} + \frac{1}{36}A_{2,3} + \frac{1}{9}A_{0,2} + \frac{1}{9}A_{1,1} + \frac{1}{9}A_{1,3} + \frac{1}{9}A_{2,2} + \frac{4}{9}A_{1,2},$$

$$P_{1,0} = \frac{1}{18}A_{2,0} + \frac{1}{18}A_{2,2} + \frac{1}{9}A_{1,0} + \frac{1}{9}A_{1,2} + \frac{2}{9}A_{2,1} + \frac{4}{9}A_{1,1},$$

$$P_{1,1} = \frac{1}{9}A_{2,2} + \frac{2}{9}A_{1,2} + \frac{2}{9}A_{2,1} + \frac{4}{9}A_{1,1},$$

$$P_{1,2} = \frac{1}{9}A_{2,1} + \frac{2}{9}A_{1,1} + \frac{2}{9}A_{2,2} + \frac{4}{9}A_{1,2},$$

$$P_{1,3} = \frac{1}{18}A_{2,1} + \frac{1}{18}A_{2,3} + \frac{1}{9}A_{1,1} + \frac{1}{9}A_{1,3} + \frac{2}{9}A_{2,2} + \frac{4}{9}A_{1,2},$$

$$P_{2,0} = \frac{1}{18}A_{1,0} + \frac{1}{18}A_{1,2} + \frac{1}{9}A_{2,0} + \frac{1}{9}A_{2,2} + \frac{2}{9}A_{1,1} + \frac{4}{9}A_{2,1},$$

$$P_{2,1} = \frac{1}{9}A_{1,2} + \frac{2}{9}A_{1,1} + \frac{2}{9}A_{2,2} + \frac{4}{9}A_{2,1},$$

$$P_{2,2} = \frac{1}{9}A_{1,1} + \frac{2}{9}A_{1,2} + \frac{2}{9}A_{2,1} + \frac{4}{9}A_{2,2},$$

$$P_{2,3} = \frac{1}{18}A_{1,1} + \frac{1}{18}A_{1,3} + \frac{1}{9}A_{2,1} + \frac{1}{9}A_{2,3} + \frac{2}{9}A_{1,2} + \frac{4}{9}A_{2,2},$$

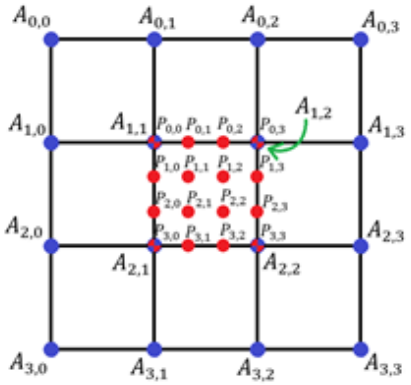
$$P_{3,0} = \frac{1}{36}A_{1,0} + \frac{1}{36}A_{1,2} + \frac{1}{36}A_{3,0} + \frac{1}{36}A_{3,2} + \frac{1}{9}A_{1,1} + \frac{1}{9}A_{2,0} + \frac{1}{9}A_{2,2} + \frac{1}{9}A_{3,1} + \frac{4}{9}A_{2,1},$$

$$P_{3,1} = \frac{1}{18}A_{1,2} + \frac{1}{18}A_{3,2} + \frac{1}{9}A_{1,1} + \frac{1}{9}A_{3,1} + \frac{2}{9}A_{2,2} + \frac{4}{9}A_{2,1},$$

$$P_{3,2} = \frac{1}{18}A_{1,1} + \frac{1}{18}A_{3,1} + \frac{1}{9}A_{1,2} + \frac{1}{9}A_{3,2} + \frac{2}{9}A_{2,1} + \frac{4}{9}A_{2,2},$$

$$P_{3,3} = \frac{1}{36}A_{1,1} + \frac{1}{36}A_{1,3} + \frac{1}{36}A_{3,1} + \frac{1}{36}A_{3,3} + \frac{1}{9}A_{1,2} + \frac{1}{9}A_{2,1} + \frac{1}{9}A_{2,3} + \frac{1}{9}A_{3,2} + \frac{4}{9}A_{2,2},$$

čia koeficientai prie $A_{i,j}$, $i, j = 0, 1, 2, 3$ atitinka tenzorinės dviejų trečios eilės Bezjė kreivių sandaugos koeficientus. Šis algoritmas taikomas tada, kai 3.13 paveiksle pavaizduoti taškai $A_{1,1}$, $A_{1,2}$, $A_{2,1}$ ir $A_{2,2}$ nėra ypatingieji, t. y. turi po 4 incidenčias briaunas. Kitu atveju šis algoritmas netinka, nes taškų skaičius, skirtas kontroliniams taškams apskaičiuoti, nelygus 16.



3.13 paveikslas. Kontrolinių Bezjė paviršiaus taškų apskaičiavimas

3.2.2 2 žingsnis. C^2 paviršiaus sudarymas ypatingajam keturkampiui

Kvadrianguliariųjų objektų viršūnės nebūtinai turi po 4 incidenčias briaunas. Tokiu atveju šios viršūnės vadinamos ypatingaisiais taškais, kurių aplinkoje sukonstruoti C^2 paviršius minėtu algoritmu negalima. Darbuose [49, 70] nagrinėjami metodai šiai problemai spręsti. Ypatingųjų taškų srityse realizuojamas Catmullo ir Clarko paviršių dalijimo algoritmas. Tokiu atveju kvadrianguliaraus paviršiaus keturkampiai, kuriems yra taikomas šis dalijimas, pakeičia savo viršūnių koordi-

nates pagal formulę:

$$v_{i+1} = \frac{1}{2}v_i + \frac{1}{16}\sum_{j=1}^{4} e_{i,j} + \frac{1}{16}\sum_{j=1}^{4} f_{i+1,j}, \tag{3.21}$$

čia $e_{i,j}$ – viršūnės, gretimos viršūnei v_i, $f_{i+1,j}$ – keturkampio, gretimo viršūnei v_i, centras. Taip pat apskaičiuojami nauji taškai, atitinkantys briaunas

$$e_{i+1,j} = \frac{1}{4}\big(v_i + e_{i,j} + f_{i+1,j-1} + f_{i+1,j}\big), \tag{3.22}$$

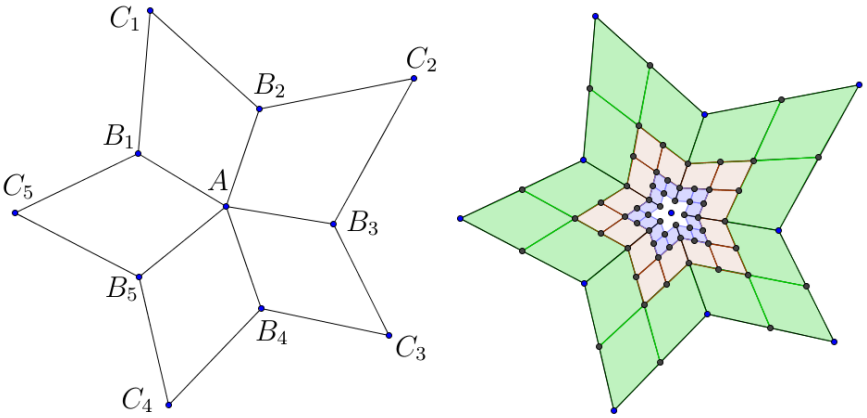
čia $f_{i+1,j-1}$ ir $f_{i+1,j}$ – gretimų briaunai $\{v_i, e_{i,j}\}$ keturkampių centrai. Jei kvadrianguliariojo paviršiaus viršūnės ir briaunos yra kraštinės, tada taikomos šios formulės naujo paviršiaus viršūnėms apskaičiuoti

$$e_{i+1,j} = \frac{1}{2}\big(v_i + e_{i,j}\big), \tag{3.23}$$

$$v_{i+1,j} = \frac{1}{8}\big(e_i + 6v_i + e_{i,j}\big). \tag{3.24}$$

Pagal (3.21), (3.22), (3.23), (3.24) lygybes kvadrianguliariajam paviršiui apskaičiavę naujų viršūnių koordinates ir sujungę gautas viršūnes briaunomis iš esmės atliekame vieną Catmullo ir Clarko algoritmo iteraciją. Kadangi Catmullo ir Clarko algoritmo rezultatas konverguoja į C^2 paviršių, o mūsų tikslas – sukonstruoti C^2 paviršių iš atskirų Bezjė paviršių, tai galima realizuoti vieną Catmullo ir Clarko iteraciją pradiniam kvadrianguliariajam paviršiui, o paskui taikyti minėtą kontrolinių taškų apskaičiavimo algoritmą. Problema – savo pozicijose išliekantys ypatingieji taškai, tačiau, paeiliui taikant Catmullo ir Clarko algoritmą ir kontrolinių taškų apskaičiavimo algoritmą, galima vis mažesniais Bezjė paviršiais konstruoti C^2 paviršių ypatingųjų taškų srityse (3.14 paveikslas). Pirmoje dalyje pateiktas ypatingasis taškas, į kurį remiasi 5 keturkampiai, antroje dalyje pavaizduotas šios srities užpildymas 3 eilės Bezjė paviršiais, taikant 3 minėto algoritmo iteracijas (skirtingų dydžių keturkampiai rodo skirtingų iteracijų rezultatus).

Kadangi 3.14 paveiksle pavaizduotas paviršių dalijimo procesas yra begalinis, praktikoje užtenka realizuoti 4–5 minėto algoritmo iteracijas, o centro tuštumą užpildyti papildoma detale, uždengiančia centre esančią tuštumą [103]. Jei toks modelis taikomas, pavyzdžiui, animaciniame filme, plika akimi tokie užpildymai nematomi. Šiame disertacijos skyriuje nagrinėjamas kitas būdas, kaip spręsti ypatingojo paviršiaus taško srities užpildymo problemą. Akivaizdu, kad aproksimuojant pradinį modelį C^2 paviršiumi galima pasirinkti norimą rezultato tikslumą,



3.14 paveikslas. Bezjė paviršių generavimas ypatingojo taško aplinkoje

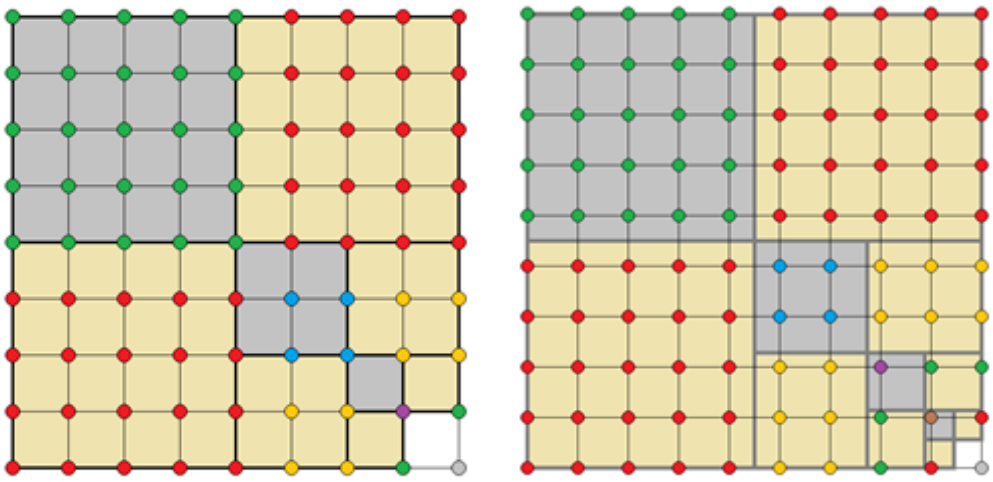
t. y. Bezjė tinklo UV tankumą. Jei šis tankumas būtų traktuojamas kaip pradinio kvadianguliariojo paviršiaus briaunoje esančių taškų skaičius, akivaizdu, kad begalinis dalijimas minėtu algoritmu nebūtų prasmingas, nes tokiu atveju egzistuotų be galo daug nereikalingų Bezjė paviršių, kurių parametrinės lygtys nebūtų reikalingos UV tinklo taškams apskaičiuoti. Todėl naudinga remtis straipsniu [77], pagal kurį centrinis ypatingasis taškas A konverguoja į tašką A'

$$A' = \frac{m}{m+5}A + \frac{4}{m(m+5)}\sum_{i=1}^{m} B_i + \frac{1}{m(m+5)}\sum_{i=1}^{m} C_i, \tag{3.25}$$

čia B_i – taškai, esantys vienos briaunos atstumu nuo ypatingojo taško A, C_i – taškai, esantys dviejų briaunų atstumu nuo ypatingojo taško A, m – ypatingojo taško A eilė (pavyzdžiui, 3.14 paveiksle $m = 5$). Likusius UV tinklo taškus galima apskaičiuoti begalinį Catmullo ir Clarko dalijimo ir vis mažesnių Bezjė paviršių formavimo procesą pakeitus baigtiniu. Pirmiausia apskaičiuokime, kiek dalijimo iteracijų turime realizuoti, jei UV tinklo kraštinėje yra n taškų:

$$z = [\log_2(n-2)] + 1. \tag{3.26}$$

Pagal (3.26) formulės natūralią reikšmę z reikia dalyti į ypatingąjį tašką besiremiantį keturkampį. Dalijimo pavyzdžiai pateikti 3.15 paveiksle. Reikia apskaičiuoti tokias kiekvieno naujo keturkampio, pažymėto 3.15 paveiksle skirtinga spalva, $u, v \in [0, 1]$ reikšmes, kurias įsistačius į (3.20) lygtį apskaičiuoti Bezjė paviršiaus taškai atitiktų UV tinklo taškus. Tai galima padaryti įgyvendinus 7 algoritmą.



(a) UV tinklas, kai $n = 9$ **(b)** UV tinklas, kai $n = 10$

3.15 paveikslas. Retasis ir baigtinis parametrinių mazgų tinklas

---

7 algoritmas. Parametrinių mazgų tinklo sudarymas

Įvesties duomenys : n

Išvesties duomenys: $\mathbf{G} = (g_{i,j}), \mathbf{H} = (h_{i,j})$

$d \leftarrow [\log_2(n-2)] + 1;$

$m_1 \leftarrow [\frac{n+2}{2}];$

$m_2 \leftarrow n + 1 - m_1;$

for $i \leftarrow 1$ ***to*** d **do**

  $p \leftarrow \frac{2^i m_2}{n} - 1;$

  for $j \leftarrow 1$ ***to*** m_1 **do**

    $g_{i,m_1-j+1} \leftarrow 1 - p - \frac{(j-1)2^i}{n};$

  end

  for $k \leftarrow 1$ ***to*** m_2 **do**

    $h_{i,m_2-k+1} \leftarrow 1 - \frac{(k-1)2^i}{n};$

  end

  $m_1 \leftarrow [\frac{m_2}{2}];$

  $m_2 \leftarrow m_2 - m_1;$

end

Pagal 3.15a paveikslą pasiūlyto algoritmo rezultatas, kai $n = 9$, lygus:

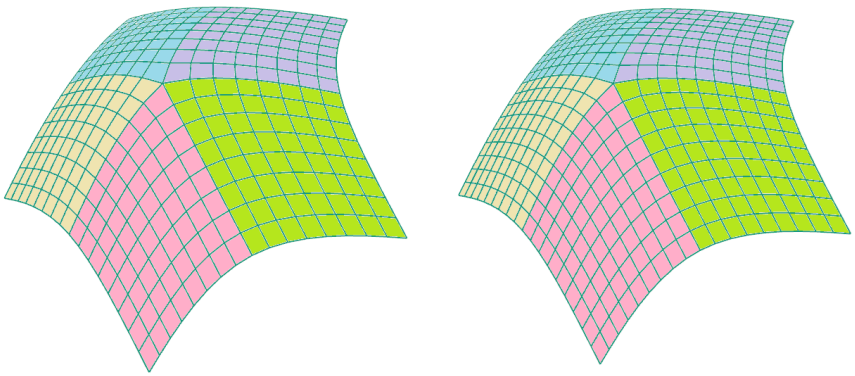
$$\mathbf{G} = \begin{pmatrix} 0 & \frac{1}{4} & \frac{1}{2} & \frac{3}{4} & 1 \\ \frac{1}{2} & 1 & - & - & - \\ 1 & - & - & - & - \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{3}{4} & 1 \\ \frac{1}{2} & 1 & - & - \\ 1 & - & - & - \end{pmatrix}.$$

Simbolis „–“ žymi neapibrėžtą matricų elementą. **G** matricos kiekviena eilutė atitinka 3.15a paveiksle kvadrato, einančio per įstrižainę, $u, v \in [0, 1]$ reikšmes, **H** matricos kiekviena eilutė analogiškai atitinka kitų iš dešinės į kairę išdėstytų kvadratų $u, v \in [0, 1]$ reikšmes.

Pagal 3.15b paveikslą pasiūlyto algoritmo rezultatas, kai $n = 10$, lygus:

$$\mathbf{G} = \begin{pmatrix} 0 & \frac{2}{9} & \frac{4}{9} & \frac{2}{3} & \frac{8}{9} \\ \frac{2}{9} & \frac{2}{3} & - & - & - \\ \frac{2}{9} & - & - & - & - \\ \frac{2}{9} & - & - & - & - \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} \frac{1}{9} & \frac{1}{3} & \frac{5}{9} & \frac{7}{9} & 1 \\ \frac{1}{9} & \frac{5}{9} & 1 & - & - \\ \frac{1}{9} & 1 & - & - & - \\ 1 & - & - & - & - \end{pmatrix}.$$

Toliau formuojant C^2 paviršių matricų **G** ir **H** elementai naudojamos $u, v \in [0, 1]$ parametrų reikšmės galutinio rezultato taškams apskaičiuoti. Šie taškai turi būti išrikiuojami ir suformuojami nauji keturkampiai. 3.16 paveiksle pateiktas išrikiuotų ir susietų taškų pavyzdys, atitinkantis 3.14 paveiksle esančią konstrukciją, kurios C^2 paviršius apskaičiuotas pagal 3.15a ir 3.15b paveikslų schemas.

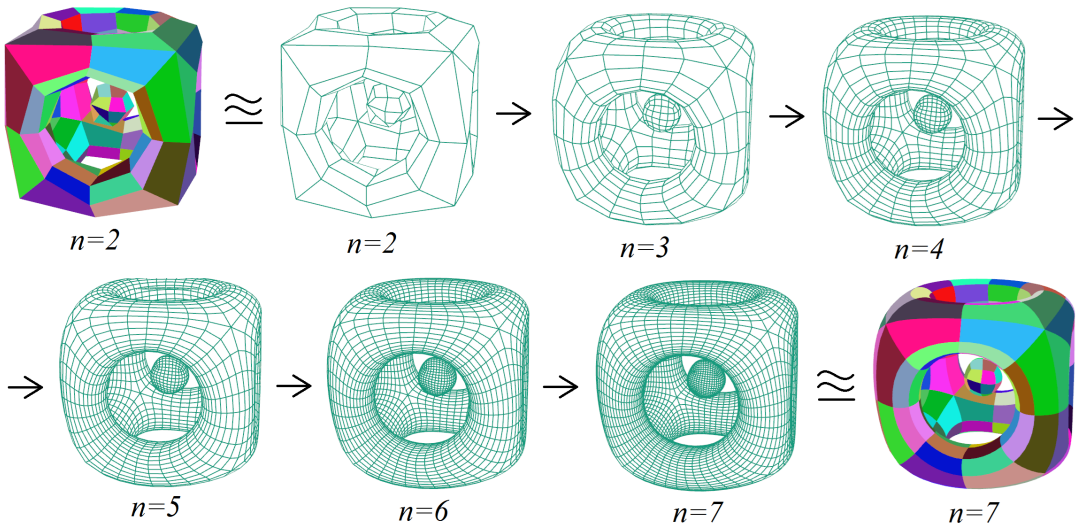


3.16 paveikslas. C^2 paviršiaus formavimas ypatingųjų taškų srityse

3.2.3 3 žingsnis. Jungtinio C^2 paviršiaus sudarymas

Pasinaudoję 1 ir 2 žingsniais, pagal pasirinktą detalumo parametrą n sudarome C^2 paviršiaus keturkampių tinklo aproksimaciją kiekvienam įvesties $T_\square$ paviršiaus keturkampiui. Belieka šiuos keturkampius tarpusavyje susieti pašalinant dubliuotas kraštines viršūnes. 3.10 paveiksle pateiktas algoritmas buvo realizuotas *Maple* programos aplinkoje. Įvesties ir išvesties duomenims pasirinktas OFF skaitmeninių modelių formatas. 3.16 paveiksle pavaizduoti galimi du algoritmo rezultatai,

atitinkantys C^2 paviršiaus aproksimaciją, kai paviršiaus detalumo parametras n kinta nuo 2 iki 7. Jei palygintume šiuos rezultatus su standartinio Catmullo ir Clarko algoritmo rezultatais, tai $n = 2$ ir $n = 4$ atvejais rezultatas būtų identiškas, kitų rezultatų (kai $n = 3, 5, 6, 7$) taikant Catmullo ir Clarko algoritmą gauti neįmanoma.

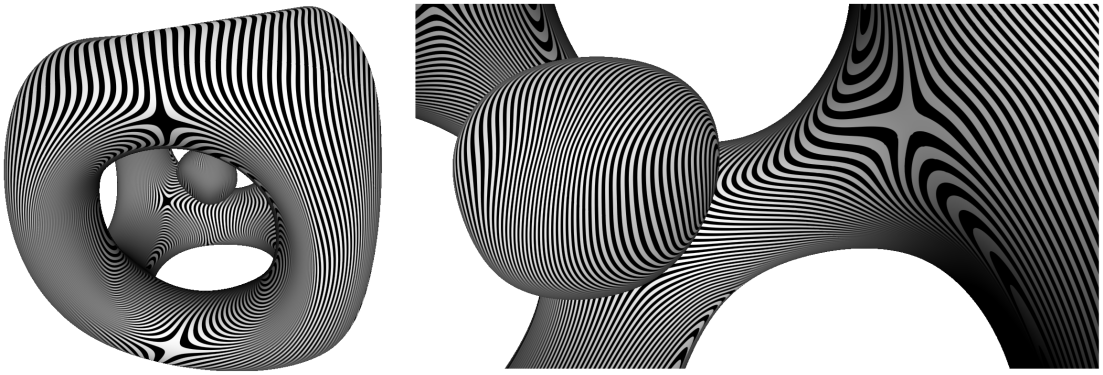


3.17 paveikslas. Paviršiaus dalijimas

Šiame darbe siūlomas paviršių dalijimo algoritmas pranašesnis už Loopo ir Schaeferio algoritmą [65] tuo, kad grąžinamas C^2 paviršius. Loopo ir Schaeferio algoritmas ypatingųjų taškų srityse generuoja Bezjė paviršius, tačiau jų sandūroje nesutampa antros eilės išvestinės. 3.18 paveiksle pateiktas 3.10 paveiksle aprašyto algoritmo rezultatas, kurio paviršius padengtas atspindžio linijomis (angl. *reflection lines*). Šios nelūžtančios linijos rodo, kad gautas rezultatas – C^2 paviršius [53]. Disertacijos prieduose pateiktas jungtinio C^2 paviršiaus sudarymo iš keturkampių tinklo paviršiaus pirminis programos tekstas, skirtas *Maple* programavimo aplinkai.

3.2.4 Skaitmeninių kurpalių sudarymo algoritmo taikymas

3.8 paveiksle pateiktas kurpalio paviršiaus modeliavimo pavyzdys. Tokio pobūdžio modeliavimas vyksta rankiniu būdu formuojant paviršių (pavyzdžiui, sudarytą iš NURBS paviršių). Tokiu būdu modeliuojant svarbu nuskenuotą žmogaus pėdos paviršių aproksimuoti kurpalio paviršiumi. Šiame skyriuje siūlomas



3.18 paveikslas. Paviršiaus atspindžio linijos

algoritmas tinkamas sukurti kompiuterinio modeliavimo programai, kuria būtų konstruojamas C^2 paviršius rankiniu būdu modeliuojant skaitmeninius 3D kurpalius.

3.3 Skyriaus išvados

Šiame skyriuje apžvelgti autoriaus siūlomi 3D skaitmeninių modelių apdorojimo algoritmai:

1. *Kurpalių skaitmeninių modelių segmentavimas.* Sukurtą naują kurpalių skaitmeninių modelių paviršių segmentavimo algoritmą pravartu išbandyti realiuose kurpalių gamybos procesuose, nes šio algoritmo rezultatas, ortopedijos ekspertų nuomone, sutampa su vienetinės avalynės gamyboje atliekamu rankiniu kurpalių paviršių dalijimu.

2. *Parametrinio C^2 paviršiaus konstravimas.* Sukurtas ir realizuotas Bezjė paviršių aproksimavimo algoritmas kvadrianguliariesiems įvesties paviršiams turi 2 pagrindinius privalumus:

   a) Ypatingųjų taškų srityse apskaičiuojamos viršūnės išsidėsto tolygiai. Tai pranašesnis rezultatas už darbuose [49, 70] nagrinėjamus algoritmus, pagal kuriuos viršūnės ypatingųjų taškų srityse tankumas auga iki begalybės.

   b) Laisvai pasirenkamas tinklo parametras $n = 1, 2, 3, \ldots$ Tai pranašesnis rezultatas už darbe [20] nagrinėjamą paviršiaus dalijimo algoritmą, kuriame tinklo parametras gali įgyti reikšmes $n = 2^1, 2^2, 2^3, \ldots$

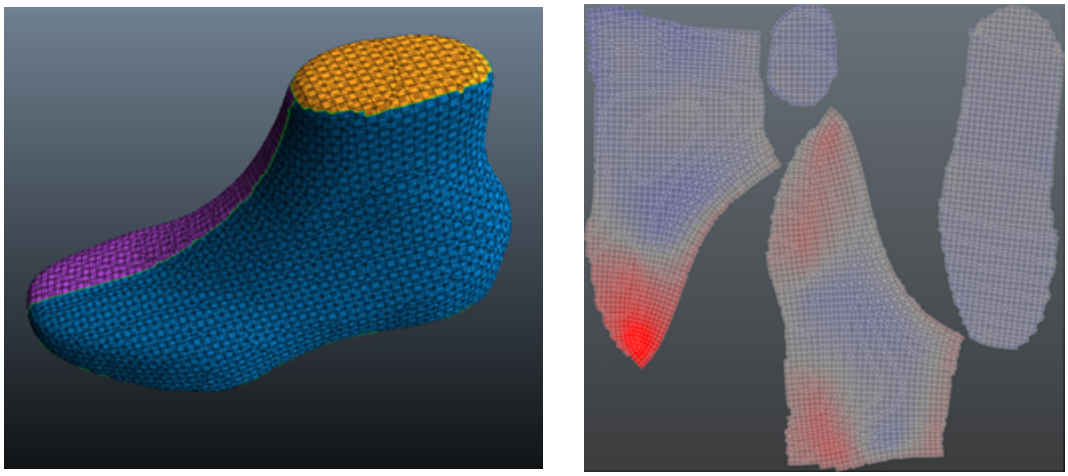
3. Disertacijoje pasiūlytas Bezjė paviršių aproksimavimo algoritmas taikytinas avalynės gamyboje kuriant (atgaminant) norimo detalumo kurpalių paviršius, kai:

    a) kompiuterio pastovioji atmintis, kurioje saugomas 3.10 paveiksle pateikto algoritmo rezultatas, turi viršutinę ribą;

    b) siekiama atgaminti norimo tikslumo modelį, pavyzdžiui, naudojant 3D spausdintuvą, papildomas C^2 paviršiaus taškų skaičius nebeturi įtakos rezultato detalumui.

4 Kurpalių paviršių išklotinių sudarymo eksperimentiniai tyrimai

Šiame skyriuje pateikiami dviejų rūšių eksperimentinių tyrimų rezultatai, skirti įvertinti paklaidoms, atsirandančioms sudarant kurpalių skaitmeninių modelių išklotines. Per pirmos rūšies eksperimentus palygintos kurpalių skaitmeninių modelių paviršiaus segmentų išklotinės, sudarytos standartiniais paviršių išklotinių sudarymo algoritmais su vienetinės avalynės gamyboje naudojamais ortopedijos ekspertų sudarytais lekalais [1A, 10A]. Šie eksperimentai aprašyti 4.1 ir 4.2 disertacijos poskyriuose atitinkamai naudojant rankinį kurpalių skaitmeninių modelių paviršiaus dalijimo metodą, aprašytą 2.3 disertacijos poskyryje ir automatinį kurpalių skaitmeninių modelių paviršiaus dalijimo metodą, pasiūlytą disertacijos 3.1 poskyryje. Kurpalių skaitmeninių modelių paviršiui dalyti rankiniu būdu buvo naudojama kompiuterinio modeliavimo programa *Blender* (4.1 paveikslas).



(a) Rankiniu būdu segmentuotas kurpalis **(b)** Segmentų išklotinės

4.1 paveikslas. Kurpalių skaitmeninių modelių apdorojimas kompiuterine modeliavimo programa *Blender*

Gauti segmentai (4.1a paveikslas) 4.1 disertacijos poskyryje buvo išplokštinti

ABF++, LSCM ir ARAP paviršių išklotinių sudarymo algoritmais. 4.1b paveiksle pateiktas kurpalio segmentų išklotinių, gautų naudojant ABF++ algoritmą, pavyzdys. Sudarytų išklotinių kontūrai buvo palyginti su etaloninių lekalų, išplokštintų naudojant gamybinę vaško foliją, kontūrais. 4.2 disertacijos poskyryje pasirinktas ARAP paviršiaus išklotinės sudarymo algoritmas dėl tiksliausių 4.1 poskyryje išklotinių palyginimo rezultatų. Taikant ARAP algoritmą ir 3.1 disertacijos poskyryje pasiūlytą automatinį kurpalių skaitmeninių modelių paviršių dalijimą 4.2 poskyryje gauti rezultatai ne blogesni nei 4.1 poskyrio rezultatai taikant ARAP algoritmą ir rankinį kurpalių paviršių dalijimą.

Antros rūšies eksperimentai 4.3 disertacijos poskyryje atlikti atskirai palyginus kiekvieno trikampių tinklui priklausančio trikampio panašumo ir ploto deformaciją, gaunamą sudarant kurpalių segmentų išklotines [9A]. Įvertinus trikampių plotus (svorius) apskaičiuotos šių deformacijų sumos.

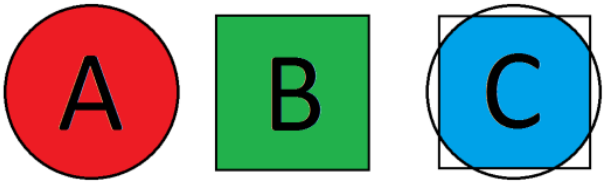
4.1 Rankiniu būdu sudarytų kurpalių segmentų išklotinių palyginimas su ekspertų sudarytomis išklotinėmis

Šio poskyrio eksperimentai atlikti disertacijoje apžvelgtais trimis pagrindiniais skaitmeninių modelių paviršių išklotinių sudarymo algoritmais ABF++, LSCM ir ARAP. Imties duomenys – 5 kurpalių poros (iš viso 10 kurpalių), kiekvieno kurpalio paviršius buvo dalijamas į 4 dalis, iš kurių išplokštinami šoniniai trikampių tinklo segmentai. Taigi iš viso buvo atlikta $5 \cdot 2 \cdot 2 = 20$ eksperimentų trimis skirtingais išklotinių sudarymo algoritmais.

Siekiant eksperimentiškai patikrinti, kaip sutampa kurpalių segmentų išplokštinimo rezultatai kurpalių paviršių dalijant *Blender* modeliavimo programa ir naudojant gamybinę vaško foliją, reikia šias išklotines uždėti vieną ant kitos taip, kad bendras susiklojimo plotas būtų didžiausias. Matematiškai tokių figūrų santykinį panašumą (4.2 paveikslas) galima apskaičiuoti pagal formulę

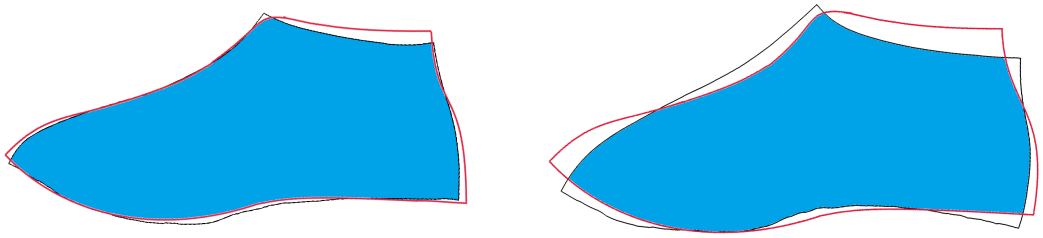
$$\omega = \frac{2C}{A+B} \cdot 100\ \%,$$

čia A ir B – tarpusavyje lyginamų figūrų plotai, C – didžiausias plotas, kurį galima gauti sudarant lyginamų figūrų sankirtą.



4.2 paveikslas. Figūrų santykinių panašumų lyginimas pagal bendrą susiklojantį plotą

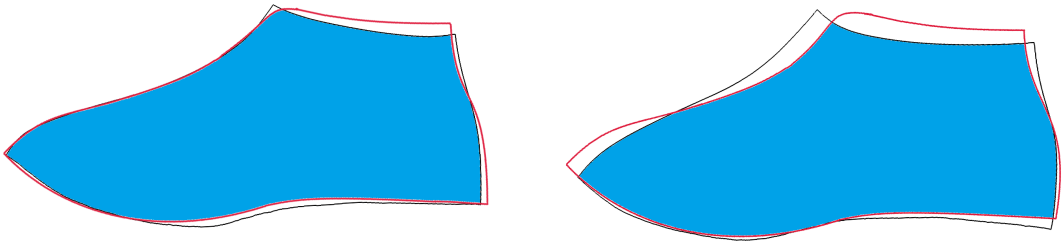
Jei $\omega = 100$ %, reiškia figūros yra lygios. Kuo ω mažesnis už 100 %, tuo figūros nepanašesnės. Pavyzdžiui, sutapatinus ploto neužimančią atkarpą ir kitą figūrą, užimančią plotą, ω būtų lygus 0, nes šios figūros išvis neturėtų susiklojančio ploto.



(a) Kairieji lekalai, $w = 98{,}23$ %

(b) Dešinieji lekalai, $w = 93{,}64$ %

4.3 paveikslas. Lekalų kontūrų lyginimas plokštinant ABF++ algoritmu

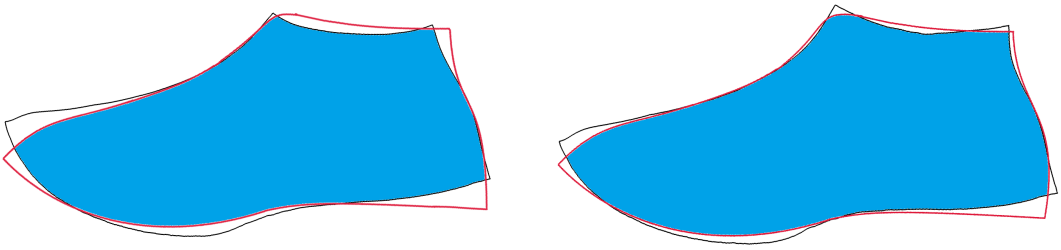


(a) Kairieji lekalai, $w = 98{,}26$ %

(b) Dešinieji lekalai, $w = 95{,}26$ %

4.4 paveikslas. Lekalų kontūrų lyginimas plokštinant LSCM algoritmu

4.3, 4.4 ir 4.5 paveiksluose lyginami gamybiniai lekalai (raudona spalva) ir lekalai, sudaryti atitinkamai ABF++, LSCM ir ARAP išplokštinimo algoritmais. Šie lekalai sudaryti pagal pirmos poros kairįjį kurpalį. Visos turimos duomenų imties išplokštinimo rezultatai pateikti 4.1, 4.2 ir 4.3 lentelėse. Šiuose lentelėse pateiktų plotų reikšmės:



(a) Kairieji lekalai, $w = 96{,}41$ %

(b) Dešinieji lekalai, $w = 98{,}13$ %

4.5 paveikslas. Lekalų kontūrų lyginimas plokštinant ARAP algoritmu

A – gamybinio lekalo, sudaryto naudojant vaško foliją, ribojamas plotas.

B – rankiniu būdu sudaryto ir ABF++, LSCM ir ARAP algoritmais išplokštinto skaitmeninio kurpalio modelio paviršiaus segmento ribojamas plotas.

C – dviejų kontūrų ribojamas didžiausias plotas, kurį galima gauti suklojus aukščiau minėtus segmentus tarpusavyje (suklojama rankiniu būdu).

Išklotinių kontūrų ribojami A, B, C plotai apskaičiuoti pagal šių kontūrų apimamą pikselių skaičių. 4.1, 4.2 ir 4.3 lentelių stulpeliuose „Atstumai“ pateikti atstumai (milimetrais) nuo kurpalio segmentų paviršiaus išklotinių iki trijų avalynės gamyboje naudojamo lekalo kritinių taškų (4.6 paveikslas). Šių kritinių taškų tikslumas itin svarbus individualios avalynės gamyboje, nes per šiuos taškus kurpalio paviršiumi einančios ašinės linijos, kurių paklaida negali viršyti 2 milimetrų, nulemia tolimesnę gamybos konstrukciją. 4.6 paveiksle atstumai iki kritinių taškų matuoti naudojant programinę įrangą *Photoshop*[10].

Atlikus išklotinių kontūrų lyginamuosius eksperimentinius tyrimus gauti šie rezultatai:

1. Naudojant ABF++ išklotinių sudarymo algoritmą vidutinis santykinis skaitmeninių ir realių lekalų panašumas 97,49 % (4.1 lentelė).

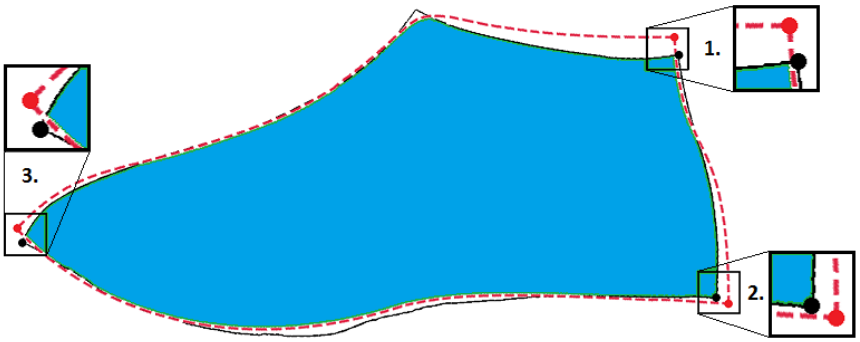
2. Naudojant LSCM išklotinių sudarymo algoritmą vidutinis santykinis skaitmeninių ir realių lekalų panašumas 97,36 % (4.2 lentelė).

3. Naudojant ARAP išklotinių sudarymo algoritmą vidutinis santykinis skaitmeninių ir realių lekalų panašumas 97,7 % (4.3 lentelė).

[10] Paveikslėlių kūrimo ir redagavimo programinė įranga *Photoshop*. Internetinė prieiga: http://www.photoshop.com.

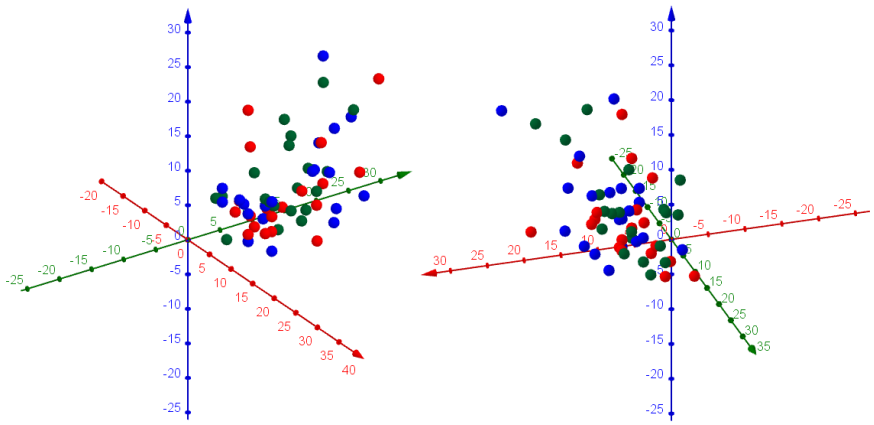
4.1 lentelė. Kurpalių išklotinių, sudarytų paviršių segmentuojant rankiniu būdu ir naudojant ABF++ išplokštinimo algoritmą, lyginimas su ortopedijos ekspertų sudarytais lekalais

Pora/kurpalis/pusė			A	B	C	$\frac{2C}{A+B}$	Atstumai
Pirmoji pora	kair.	kair.	578734	558968	558795	98,23 %	(6, 4, 7)
	kurp.	deš.	570831	549012	524288	93,63 %	(16, 10, 18)
	deš.	kair.	584171	566157	565824	98,37 %	(12, 5, 7)
	kurp.	deš.	574593	556319	547799	96,87 %	(3, 19, 4)
Antroji pora	kair.	kair.	632804	619086	613302	97,98 %	(4, 12, 3)
	kurp.	deš.	637810	609969	603171	96,67 %	(10, 23, 21)
	deš.	kair.	638950	611153	600192	96,02 %	(13, 9, 10)
	kurp.	deš.	632804	613351	607965	97,57 %	(16, 16, 12)
Trečioji pora	kair.	kair.	550236	534187	527929	97,36 %	(6, 16, 3)
	kurp.	deš.	539557	522001	521194	98,19 %	(5, 4, 5)
	deš.	kair.	547489	533744	532366	98,47 %	(2, 3, 6)
	kurp.	deš.	539960	519292	512104	96,69 %	(21, 6, 7)
Ketvirtoji pora	kair.	kair.	591323	573492	566721	97,30 %	(7, 5, 5)
	kurp.	deš.	584236	562025	559188	97,56 %	(7, 5, 15)
	deš.	kair.	588692	573845	571064	98,24 %	(9, 6, 3)
	kurp.	deš.	582918	564142	555682	96,88 %	(8, 4, 21)
Penktoji pora	kair.	kair.	505229	514098	505060	99,09 %	(5, 7, 2)
	kurp.	deš.	513236	502834	500893	98,59 %	(5, 9, 4)
	deš.	kair.	511584	508920	496786	97,36 %	(9, 7, 3)
	kurp.	deš.	508042	507705	501837	98,81 %	(8, 4, 3)
					Vidurkis:	97,49 %	(9, 9, 8)



4.6 paveikslas. Lekalų kontūrų lyginimas pagal plotus ir kritinius taškus (1, 2, 3)

4.7 paveiksle raudona, žalia ir mėlyna spalvomis atitinkamai pavaizduotos 4.1, 4.2 ir 4.3 lentelių stulpelių „Atstumai“ reikšmės trimatėje erdvėje. Iš šių atsitiktinai išsidėsčiusių taškų matome, kad santykinio lekalų panašumo rezultatai nėra reikšminiai. Atsitiktinis taškų išsidėstymas patvirtina faktą, kad eksperimentams atlikti buvo naudojami etaloniniai gamybiniai brėžiniai, t. y. suvidurkinti kurpa-



4.7 paveikslas. Susietų su euklidinėmis koordinatėmis atstumų iki kritinių taškų pasiskirstymas trimatėje erdvėje

lio kairiosios ir dešiniosios pusės paviršiaus išklotinių kontūrai.

Nors pagal turimus statistinius duomenis – ortopedijos ekspertų suvidurkintus gamybinių brėžinių kontūrus – neįmanoma atkurti dviejų pirminių originalių išplokštintos vaško folijos kontūrų, tačiau atlikti eksperimentai leidžia teigti, kad izometrinio ARAP algoritmo rezultatas nežymiai geresnis už konforminių ABF++, LSCM algoritmų rezultatus lyginant šiais algoritmais sudarytas paviršių išklotines su juos atitinkančiais etaloniniais kurpalių lekalais.

Kadangi disertacijoje buvo lyginami suvidurkinti ortopedijos ekspertų kontūrai su nesuvidurkintais eksperimentiniais tyrimais apskaičiuotais išklotinių kontūrais, paklaidos tarp kritinių taškų gautos per didelės (vidutiniškai 8–9 milimetrai), nes gamyboje galima daugiausia 2 milimetrų paklaida. Tai leidžia teigti, kad ortopedijos ekspertų atliekamas kontūrų suvidurkinimas paslenka kritinius taškus.

Ateities eksperimentiniai tyrimai susidėtų iš kurpalių skaitmeninių segmentų išklotinių kontūrų, gautų konforminiais ar izometriniais metodais, suvidurkinimo pagal apskaičiuotus įtvirčio taškus. Tačiau šiems eksperimentams atlikti pirma reikia sukurti algoritmą, skirtą kurpalių skaitmeninių modelių įtvirčio taškams nustatyti.

Kitame disertacijos poskyryje bus atliekami analogiški kurpalių lekalų kontūrų lyginamieji eksperimentai kurpalių paviršių dalijant 3.1 disertacijos poskyryje pasiūlytu algoritmu.

4.2 lentelė. Kurpalių išklotinių, sudarytų paviršių segmentuojant rankiniu būdu ir naudojant LSCM išplokštinimo algoritmą, lyginimas su ortopedijos ekspertų sudarytais lekalais

Pora/kurpalis/pusė			A	B	C	$\frac{2C}{A+B}$	Atstumai
Pirmoji pora	kair.	kair.	578734	566037	562451	98,26 %	(2, 4, 6)
	kurp.	deš.	570831	557622	537470	95,25 %	(9, 6, 8)
	deš.	kair.	584171	572527	568130	98,23 %	(8, 13, 4)
	kurp.	deš.	574593	564510	557775	97,93 %	(12, 8, 7)
Antroji pora	kair.	kair.	632804	610330	605732	97,45 %	(7, 17, 8)
	kurp.	deš.	637810	616714	614248	97,92 %	(9, 8, 3)
	deš.	kair.	638950	618928	611609	97,24 %	(3, 13, 15)
	kurp.	deš.	632804	613583	602168	96,62 %	(6, 16, 5)
Trečioji pora	kair.	kair.	550236	534587	532513	98,17 %	(10, 6, 8)
	kurp.	deš.	539557	523225	499516	94,00 %	(15, 11, 26)
	deš.	kair.	547489	532741	527805	97,72 %	(12, 9, 10)
	kurp.	deš.	539960	518497	509697	96,30 %	(9, 10, 16)
Ketvirtoji pora	kair.	kair.	591323	579351	577389	98,64 %	(8, 8, 6)
	kurp.	deš.	584236	562802	556486	97,03 %	(16, 5, 19)
	deš.	kair.	588692	574206	557751	95,92 %	(7, 13, 2)
	kurp.	deš.	582918	562587	552530	96,46 %	(22, 11, 25)
Penktoji pora	kair.	kair.	505229	504725	499029	98,82 %	(2, 9, 8)
	kurp.	deš.	513236	508851	501312	98,09 %	(13, 10, 13)
	deš.	kair.	511584	514149	507504	98,95 %	(2, 3, 6)
	kurp.	deš.	508042	505876	498177	98,26 %	(6, 2, 2)
					Vidurkis:	97,36 %	(9, 9, 10)

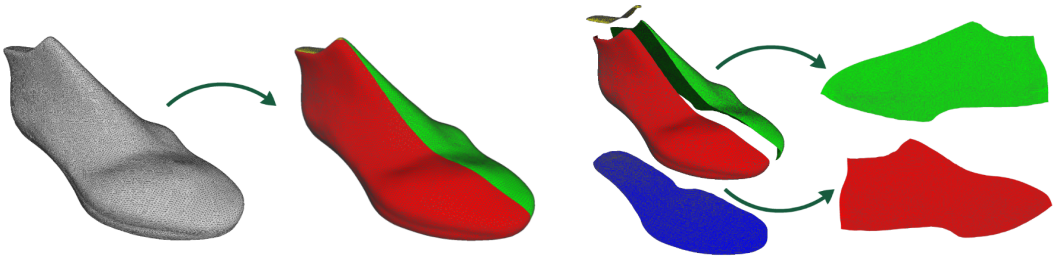
4.2 Automatiškai sudarytų kurpalių segmentų išklotinių palyginimas su ekspertų sudarytomis išklotinėmis

Šiame poskyryje apžvelgiami automatiniu segmentavimo metodu ir ARAP išklotinių sudarymo algoritmu gauti lekalai (4.8 paveikslas). Atlikto tyrimo rezultatai publikuoti darbe [9A]. ARAP algoritmas daugelio autorių pripažintas tinkamiausiu išklotinėms sudaryti dėl tiksliausių eksperimentiniu būdu sudarytų išklotinių [10A]. Šio algoritmo tinkamumą rodo ir straipsnio [62] rezultatai, pagal kuriuos ARAP algoritmas lyginamas su populiariausiais ir dažniausiais kompiuterinėse programose išklotinių sudarymo algoritmais.

Iš 4.3 ir 4.4 lentelių matome, kad vidutiniai santykiniai lekalų panašumai lygūs atitinkamai 97,7 % ir 97,85 %.

4.3 lentelė. Kurpalių išklotinių, sudarytų paviršių segmentuojant rankiniu būdu ir naudojant ARAP išplokštinimo algoritmą, lyginimas su ortopedijos ekspertų sudarytais lekalais

Pora/kurpalis/pusė			A	B	C	$\frac{2C}{A+B}$	Atstumai
Pirmoji pora	kair.	kair.	578734	564603	551139	96,40 %	(17, 16, 9)
	kurp.	deš.	570831	563310	556483	98,13 %	(13, 14, 4)
	deš.	kair.	584171	570637	559070	96,82 %	(9, 6, 7)
	kurp.	deš.	574593	569054	557354	97,46 %	(9, 16, 9)
Antroji pora	kair.	kair.	632804	632033	625012	98,82 %	(4, 17, 7)
	kurp.	deš.	637810	612580	612370	97,94 %	(14, 10, 13)
	deš.	kair.	638950	611928	603507	96,49 %	(15, 3, 11)
	kurp.	deš.	632804	613421	612873	98,35 %	(12, 5, 2)
Trečioji pora	kair.	kair.	550236	530668	527524	97,60 %	(8, 4, 2)
	kurp.	deš.	539557	524405	521426	98,01 %	(12, 13, 28)
	deš.	kair.	547489	534803	534732	98,81 %	(7, 7, 4)
	kurp.	deš.	539960	517571	499708	94,50 %	(17, 14, 21)
Ketvirtoji pora	kair.	kair.	591323	574939	566718	97,18 %	(5, 2, 7)
	kurp.	deš.	584236	566640	566392	98,42 %	(18, 11, 9)
	deš.	kair.	588692	576627	568218	97,52 %	(25, 6, 25)
	kurp.	deš.	582918	561662	552297	96,50 %	(9, 2, 9)
Penktoji pora	kair.	kair.	505229	505701	498905	98,70 %	(8, 4, 6)
	kurp.	deš.	513236	511408	505028	98,57 %	(11, 13, 15)
	deš.	kair.	511584	508521	504457	98,90 %	(7, 4, 7)
	kurp.	deš.	508042	506398	501156	98,80 %	(5, 2, 9)
					Vidurkis:	97,70 %	(11, 8, 10)



(a) Naujas automatinis segmentavimas **(b)** Išplokštinimas ARAP algoritmu

4.8 paveikslas. Kurpalių skaitmeninių modelių automatinis segmentavimas ir plokštinimas

Tai leidžia teigti, kad disertacijoje pasiūlytu automatiniu būdu sudaryti segmentai yra tikslesni, t. y. labiau atitinka ekspertų atliekamą kurpalių paviršiaus dalijimą.

Taip pat tai patvirtina ir mažesni atstumai iki kritinių taškų. Siekiant gauti dar tikslesnius išklotinių deformacijų rezultatus kitame poskyryje aprašomi atlikti antros rūšies eksperimentai – apskaičiuotos kiekvienam trikampių tinklo paviršiui

4.4 lentelė. Kurpalių išklotinių, sudarytų paviršių segmentuojant automatiniu metodu ir naudojant ARAP išplokštinimo algoritmą, lyginimas su ortopedijos ekspertų sudarytais lekalais

Pora/kurpalis/pusė			A	B	C	$\frac{2C}{A+B}$	Atstumai
Pirmoji pora	kair.	kair.	578734	560992	558821	98,06 %	(14, 8, 8)
	kurp.	deš.	570831	552902	549576	97,81 %	(15, 9, 5)
	deš.	kair.	584171	566562	565360	98,26 %	(7, 8, 5)
	kurp.	deš.	574593	560913	556637	98,04 %	(10, 11, 8)
Antroji pora	kair.	kair.	632804	620209	613862	97,98 %	(4, 17, 7)
	kurp.	deš.	637810	613390	605394	96,77 %	(7, 8, 2)
	deš.	kair.	638950	615940	610795	97,35 %	(16, 4, 4)
	kurp.	deš.	632804	614362	612701	98,25 %	(17, 7, 2)
Trečioji pora	kair.	kair.	550236	538670	531339	97,59 %	(9, 4, 3)
	kurp.	deš.	539557	525270	515685	96,86 %	(10, 11, 17)
	deš.	kair.	547489	530081	523175	97,10 %	(6, 7, 4)
	kurp.	deš.	539960	527116	521127	97,67 %	(14, 11, 13)
Ketvirtoji pora	kair.	kair.	591323	576057	567865	97,29 %	(2, 5, 7)
	kurp.	deš.	584236	573207	561424	97,01 %	(15, 8, 9)
	deš.	kair.	588692	575608	573699	98,55 %	(19, 6, 21)
	kurp.	deš.	582918	554788	553593	97,32 %	(10, 3, 12)
Penktoji pora	kair.	kair.	505229	510233	501036	98,68 %	(9, 2, 2)
	kurp.	deš.	513236	508059	505466	98,99 %	(7, 9, 12)
	deš.	kair.	511584	499728	498055	98,50 %	(8, 5, 9)
	kurp.	deš.	508042	506635	501354	98,82 %	(4, 2, 5)
					Vidurkis:	97,85 %	(10, 7, 7)

priklausančio trikampio deformacijos bei jų sumos pagal svorius, kuriems priskirti trikampių plotai.

4.3 Sudarytų kurpalių segmentų išklotinių deformacijos įverčiai

Šiame poskyryje aprašomi darbe [7A] atlikti eksperimentai, kuriais buvo apskaičiuotos kurpalių segmentų išklotinių kampų ir ploto deformacijos D_{pan}, D_{plot} pagal formules (2.19), (2.20), kai $\theta = 1$:

$$D_{pan} = \frac{1}{\sum_{i=1}^{T} S_i} \sum_{j=1}^{T} S_j \Big(\frac{\sigma_{1,1}^j}{\sigma_{2,2}^j} + \frac{\sigma_{2,2}^j}{\sigma_{1,1}^j} \Big),$$

$$D_{plot} = \frac{1}{\sum_{i=1}^{T} S_i} \sum_{j=1}^{T} S_j \Big(\sigma_{1,1}^j \sigma_{2,2}^j + \frac{1}{\sigma_{1,1}^j \sigma_{2,2}^j} \Big).$$

Čia $\sigma^j_{1,1}$ ir $\sigma^j_{2,2}$ – j-ojo trikampių tinklui priklausančio trikampio transformacijos iš $T_\triangle$ į $\overline{T}_\triangle$ 2×2 Jakobio matricos singuliariosios matricų $\mathbf{U\Sigma V}$ dekompozicijos matricos $\mathbf{\Sigma}$ įstrižainės elementai.

4.5 lentelė. Kurpalių išklotinių, sudarytų paviršių segmentuojant automatiniu metodu ir naudojant ARAP išplokštinimo algoritmą, deformacijos rezultatai

Pora/kurpalis/pusė			D_{pan}	D_{plot}
Pirmoji pora	kairysis kurpalis	kairioji	2,002279	2,001967
		dešinioji	2,003052	2,002540
	dešinysis kurpalis	kairioji	2,002500	2,001898
		dešinioji	2,003055	2,002652
Antroji pora	kairysis kurpalis	kairioji	2,004012	2,003321
		dešinioji	2,003033	2,002514
	dešinysis kurpalis	kairioji	2,002495	2,001941
		dešinioji	2,002348	2,001743
Trečioji pora	kairysis kurpalis	kairioji	2,005540	2,004863
		dešinioji	2,002748	2,002149
	dešinysis kurpalis	kairioji	2,003166	2,002556
		dešinioji	2,002752	2,002210
Ketvirtoji pora	kairysis kurpalis	kairioji	2,001796	2,001372
		dešinioji	2,003222	2,002733
	dešinysis kurpalis	kairioji	2,003259	2,002779
		dešinioji	2,002361	2,001922
Penktoji pora	kairysis kurpalis	kairioji	2,003482	2,002879
		dešinioji	2,003054	2,002497
	dešinysis kurpalis	kairioji	2,003262	2,002575
		dešinioji	2,003816	2,003116
Vidurkis:			2,003062	2,002511

$D_{pan} \in [2; \infty)$ įvertis skaitine verte parodo, kiek deformuojamas išklotinės suminis plotas pagal kiekvieno trikampio kampų pokyčius, $D_{plot} \in [2; \infty)$ įvertis skaitine verte parodo, kiek deformuojamas išklotinės suminis plotas pagal kiekvieno trikampio ploto pokyčius. Šie įverčiai apskaičiuoti visų 20 kurpalių segmentų, kurie išplokštinti ARAP, ABF++ ir LSCM algoritmais (4.5, 4.6, 4.7 lentelės).

Palyginus antros rūšies kurpalių segmentų išklotinių deformacijas gauti šie rezultatai:

1. Naudojant ARAP išklotinių sudarymo algoritmą (4.5 lentelė) deformacijų įverčiai lygūs $D_{pan} = 2{,}003062,\ D_{plot} = 2{,}002511$.

2. Naudojant ABF++ išklotinių sudarymo algoritmą (4.6 lentelė) deformacijų įverčiai lygūs $D_{pan} = 228{,}494187,\ D_{plot} = 153{,}819066$.

4.6 lentelė. Kurpalių išklotinių, sudarytų paviršių segmentuojant automatiniu metodu ir naudojant ABF++ išplokštinimo algoritmą, deformacijos rezultatai

Pora/kurpalis/pusė			D_{pan}	D_{plot}
Pirmoji pora	kairysis kurpalis	kairioji	67,484365	44,537293
		dešinioji	193,846912	196,955791
	dešinysis kurpalis	kairioji	100,659127	44,910624
		dešinioji	343,522222	263,496879
Antroji pora	kairysis kurpalis	kairioji	99,837567	56,973643
		dešinioji	111,794382	52,924877
	dešinysis kurpalis	kairioji	69,269672	37,256868
		dešinioji	85,856008	50,335862
Trečioji pora	kairysis kurpalis	kairioji	643,615669	617,980488
		dešinioji	472,906668	426,438659
	dešinysis kurpalis	kairioji	424,320681	176,170472
		dešinioji	413,775311	624,164287
Ketvirtoji pora	kairysis kurpalis	kairioji	143,989432	58,730765
		dešinioji	275,071743	64,316842
	dešinysis kurpalis	kairioji	194,836874	61,129507
		dešinioji	256,062959	85,194702
Penktoji pora	kairysis kurpalis	kairioji	128,612760	59,850476
		dešinioji	169,290929	58,276623
	dešinysis kurpalis	kairioji	239,515468	47,491096
		dešinioji	135,614988	49,245572
Vidurkis:			228,494187	153,819066

3. Naudojant LSCM išklotinių sudarymo algoritmą (4.7 lentelė) deformacijų įverčiai lygūs $D_{pan} = 207{,}7825014,\ D_{plot} = 114{,}272624$.

Pagal vidutinius deformacijų skaitinius rezultatus, lyginant su ABF++ ir LSCM algoritmų rezultatais, ARAP algoritmas besąlygiškai geriausiai minimizuoja konforminio D_{pan} ir autalinio parametrizavimo deformaciją D_{plot}, todėl kiek galima išlaikomas kiekvieno trikampio standumas (kraštinių ilgiai). Konforminių išklotinių sudarymo algoritmų ABF++ ir LSCM rezultatai – didelės skaitinės vertės deformacijos įverčiai – parodo, kad, išlaikant trikampių panašumą, labiausiai buvo deformuoti trikampių plotai, todėl, naudojant šiuos algoritmus vienetinės avalynės gamyboje, oda gali būti per daug įtempta ir susiraukšlėjusi labiausiai deformuotose išklotinių srityse.

4.7 lentelė. Kurpalių išklotinių, sudarytų paviršių segmentuojant automatiniu būdu ir naudojant LSCM išplokštinimo algoritmą, deformacijos rezultatai

Pora/kurpalis/pusė			D_{pan}	D_{plot}
Pirmoji pora	kairysis kurpalis	kairioji	73,658131	47,068532
		dešinioji	178,145715	41,523618
	dešinysis kurpalis	kairioji	131,561724	53,341252
		dešinioji	74,766342	48,817117
Antroji pora	kairysis kurpalis	kairioji	101,898985	85,386307
		dešinioji	100,318110	129,671260
	dešinysis kurpalis	kairioji	70,823712	35,467730
		dešinioji	74,265276	40,583627
Trečioji pora	kairysis kurpalis	kairioji	432,563824	242,598516
		dešinioji	329,351290	333,720500
	dešinysis kurpalis	kairioji	308,979176	349,216989
		dešinioji	706,686313	252,617066
Ketvirtoji pora	kairysis kurpalis	kairioji	180,266001	142,806736
		dešinioji	202,323827	65,462641
	dešinysis kurpalis	kairioji	192,958282	84,466610
		dešinioji	217,536888	100,518603
Penktoji pora	kairysis kurpalis	kairioji	159,243753	52,875238
		dešinioji	185,676280	53,013034
	dešinysis kurpalis	kairioji	313,477931	82,379417
		dešinioji	121,148468	43,917686
Vidurkis:			207,782501	114,272624

4.4 Skyriaus išvados

1. Atlikus eksperimentus buvo pastebėta, kad vidutinis santykinis lekalų panašumas 97,85 %, o tai rodo, kad disertacijoje pasiūlytu automatiniu kurpalių paviršių dalijimo algoritmu ir izometriniu ARAP išklotinių sudarymo algoritmu gautas rezultatas labiausiai atitinka ortopedijos ekspertų darbo rezultatus.

2. Disertacijoje apskaičiuoti vidutiniai skaitiniai dydžiai D_{pan} ir D_{plot} rodo, kad ARAP algoritmas absoliučiai minimizuoja trikampių kampų ir ploto deformacijas, todėl išlaikomas kiekvieno trikampio standumas. Taikant konforminius paviršiaus išklotinės sudarymo ABF++ ir LSCM algoritmus santykinės trikampių plotų deformacijos daug didesnės.

5 Bendrosios išvados

1. Pasiūlyta nauja technologija, kuri skirta kurpalių skaitmeninių modelių paviršių dalijimui (segmentavimui). Pagal šią technologiją automatiškai sudaromi kurpalių paviršių segmentai. Juos išplokštinus ARAP algoritmu gaunamas 97,85 % atitikimas su gamyboje naudojamais lekalais.

2. Eksperimentiškai nustatyta, kad izometrinio tipo paviršiaus išklotinių sudarymo ARAP algoritmo rezultatai, lyginant su ABF++ ir LSCM algoritmų gautais rezultatais, labiausiai atitinka sudaromas išklotines vienetinės avalynės gamyboje naudojant vaško foliją.

3. Atlikus eksperimentus paaiškėjo, kad, taikant izometrinį trikampių tinklo išplokštinimo ARAP algoritmą kurpalių segmentų išklotinėms sudaryti, kiekvieno trikampio panašumo ir ploto deformacija minimizuojama beveik iki minimalaus dvejeto: $D_{pan} = 2{,}003062$, $D_{plot} = 2{,}002511$. Tai keliasdešimt kartų geresnis rezultatas už rezultatus, gautus, taikant konforminius ABF++ ir LSCM algoritmus.

4. Darbe pasiūlytas kvadrianguliariųjų paviršių aproksimavimo Bezjė paviršiais algoritmas pasižymi šiomis savybėmis:

   a) Ypatingųjų taškų aplinkose išvesties skaitmeninio modelio viršūnės visada išdėstomos tolygiai.

   b) Tinklo parametrą n galima laisvai pasirinkti iš aibės $\mathbb{N}$, o gerai žinomo Catmullo ir Clarko paviršiaus dalijimo algoritmo parametras n gali įgyti reikšmes $2^1, 2^2, 2^3, \ldots$

5. Pasiūlyti analitiniai 2×2 matricų skaidymo singuliariosiomis reikšmėmis sprendiniai tinka sudaryti dekompozicijai $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}$, kai matrica $\mathbf{A}$ yra neišsigimusioji, išsigimusioji arba nulinė. Taikant disertacijoje išvestus analitinius $\sigma_{1,1}$ ir $\sigma_{2,2}$ sprendinius, paviršiaus suminės deformacijos D_{pan} ir D_{plot} apskaičiuojamos daug greičiau, nes nebereikia apskaičiuoti matricų $\mathbf{U}$ ir $\mathbf{V}$.

Literatūra

[1] Agathos A., Pratikakis I., Perantonis S., Sapidis N., Azariadis P. *3D mesh segmentation methodologies for CAD applications*, 2007.

[2] Alliez P., Bala K., Zhou K. B-mesh: A fast modeling system for base meshes of 3D articulated shapes. *Pacific Graphics* **29** (2010), no. 7, 2169–2177.

[3] Alliez P., Colin de Verdière É., Devillers O., Isenburg M. Isotropic surface remeshing. *Proc. of Shape Modeling International 03* (2003), 49–58.

[4] Alliez P., Ucelli G., Gotsman C., Attene M. Recent advances in remeshing of surfaces. *Shape analysis and structuring* (2008), 53–82.

[5] Amenta N., Bern M., Eppstein D. Optimal point placement for mesh smoothing. *Journal of Algorithms* **30** (1999), no. 2, 302–322.

[6] Barequet G., Kumar S. Repairing CAD models. *Visualization'97., Proceedings* (1997), 363–370.

[7] Barequet G., Sharir M. Filling gaps in the boundary of a polyhedron. *Computer-Aided Geometric Design* **12** (1995), no. 2, 207–229.

[8] Benhabiles H., Vandeborre J., Lavoue G., Daoudi M. A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3D-models. *Shape Modeling and Applications, 2009. SMI 2009.* IEEE International Conference on (2009), 36–43.

[9] Bernardini F., Mittleman M., Rushmeier H., Silva C., Taubin G. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics* **5** (1999), no. 4, 349–359.

[10] Bischoff S. Kobbelt L. Structure preserving cad model repair. *Computer Graphics Forum (Proc. Eurographics 05)* **24** (2005), no. 3, 527–536.

[11] Bischoff S., Pavic D., Kobbelt L. Automatic restoration of polygon models. *Transactions on Graphics* **24** (2005), no. 4, 1332–1352.

[12] Blinn J. Consider the lowly 2 × 2 matrix. *IEEE Computer Graphics and Applications* **16** (1996), no. 5, 82–88.

[13] Boissonnat J. D. Geometric structures for three dimensional shape representation. *ACM Transactions on Graphics (TOG)* **3** (1984), no. 4, 266–286.

[14] Boissonnat J. D., Oudot S. Provably good sampling and meshing of surfaces. *Graphical Models* **67** (2005), 405–451.

[15] Borodin P., Novotni M., Klein R. Progressive gap closing for mesh repairing. *Advances in Modelling, Animation and Rendering* (2002), 201–213.

[16] Botsch M., Kobbelt L., Pauly M., Pierre A., Lévy B. *Polygon mesh processing*, CRC Press, 2010.

[17] Botsch M., Pauly M., Gross M., Kobbelt L. Primo: Coupled prisms for intuitive surface modeling. *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2006), 11–20.

[18] Burchard H. G., Ayers J. A., Frey W. H., Sapidis N. S. Approximation with aesthetic constraints. *In Designing Fair Curves and Surfaces* (1994), 3–28.

[19] Carbal B., Leedom L. C. Imaging vector fields using line integral convolution. *SIGGRAPH 93 Conference Proceedings* (1993), 263–274.

[20] Catmull E., Clark J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* **10** (1978), no. 6, 350–355.

[21] Chen L. Mesh smoothing schemes based on optimal Delaunay triangulations. *Proceedings of 13th International Meshing Roundtable* (2004), 109–120.

[22] Chen X., Golovinskiy A., Funkhouser T. *A benchmark for 3D mesh segmentation*, ACM Transactions on Graphics (TOG) **28** (2009), no. 3.

[23] Chung F. *Spectrac graph theory*, American Mathematical Society, 1997.

[24] Curless B., Levoy M. A volumetric method for building complex models from range images. *Proc. of ACM SIGGRAPH 96* (1996), 303–312.

[25] Davidsonas A. *Parametrinio paviršiaus modelio sudarymas iš trimačio taškų debesies (daktaro disertacija)*, Vytauto Didžiojo universitetas, 2013.

[26] Davis J., Marschner S., Garr M., Levoy M. Filling holes in complex surfaces using volumetric diffusion. *Proc. International Symposium on 3D Data Processing, Visualization, Transmission* (2002), 428–438.

[27] Degener P., Meseth J., Klein R. An adaptable surface parameterization method. *In Proc. of 12th International Meshing Roundtable* (2003), 227–237.

[28] Dey T., Li G., Ray T. Polygonal surface remeshing with Delaunay refinement. *14th Intl. Meshing Roundtable* (2005), 343–361.

[29] Demmel J. W., Eisenstat S. C., Gilbert J. R., Li X. S., Liu J. W. H. A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications* **20** (1999), no. 3, 720–755.

[30] Desbrun M., Meyer M., Schröder P., Barr A. H. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proc. of ACM SIGGRAPH 99* (1999), 317–324.

[31] Dijkstra E. W. A note on two problems in connexion with graphs. *Numerische Mathematik* **1** (1959), 269–271.

[32] Dolzhenko E. P. *Conformal mapping*, Springer, 2001.

[33] Dong S., Bremer P.-T., Garland M., Pascucci V., Hart J. C. Spectral mesh quadrangulation. *ACM Transactions on Graphics (SIGGRAPH 2006 special issue)* (2006).

[34] Du Q., Faber V., Gunzburger M. Centroidal Voronoi tesselations: Applications and algorithms. *SIAM review* **41** (1999), no. 4, 637–676.

[35] Eppstein D. Global optimization of mesh quality. *Tutorial at the 10th Int. Meshing Roundtable*, Newport Beach, 2001.

[36] Floater M. S., Hormann K. Surface parameterization: a tutorial and survey. *Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization* (2005), 157–186.

[37] Goonetilleke, R. S. *The science of footwear*, CRC Press, 2012.

[38] Gortler S. J., Gotsman C., Thurston D. *One-forms on meshes and applications to 3D mesh parameterization*, Tech. Rep. CS TR-12-04, Harvard University, 2004.

[39] Greß A., Klein R. Efficient representation and extraction of 2-manifold isosurfaces using kd-trees. *Proc. 11th Pacific Conference on Computer Graphics and Applications (PG 2003)* (2003), 364–376.

[40] Guéziec A., Taubin G., Lazarus F., Horn B. Cutting and stitching: Converting sets of polygons to manifold surfaces. *IEEE Transactions on Visualization and Computer Graphics* **7** (2001), no. 2, 136–151.

[41] Guskov I., Wood Z. J. Topological noise removal. *Proc. of Graphics Interface 2001* (2001), 19–26.

[42] Hagen H., Hahmann S., Schreiber T., Nakayima Y., Wördenweber B., Hollemann-Grundstedt P. Surface interrogation algorithms. *IEEE Computer Graphics and Applications* **12** (1992), no. 5, 53–60.

[43] Haralick R. M., Sternberg S. R., Zhuang X. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **9** (1987), no. 4, 532–550.

[44] Hormann K., Greiner G. Mips: An efficient global parametrization method. *Curve and Surface Design*, Saint-Malo 1999 (2000), 153–162.

[45] Jarnik V. O jistém problému minimálním [about a certain minimal problem]. *Práce Moravské Přírodovědecké Společnosti (in Czech)* **6** (1930), 57–63.

[46] Jiao X., Bayyana N. R. Identification of C1 and C2 discontinuities for surface meshes in CAD. *Computer-Aided Design* **40** (2008), no. 2, 160–175.

[47] Karčiauskas K., Peters J. Guided C2 spline surfaces with v-shaped tessellation. *Lecture notes in computer science. ISSN: 0302-9743. 2007, N 4647: Mathematics of surfaces XII*, 12th IMA international conference, Sheffield, UK, September 4–6, 2007 proceeding (2007), 233–244.

[48] Karčiauskas K., Peters J. Lens-shaped surfaces and c-2 subdivision. *Computing* **86** (2009), no. 2-3, 171–173.

[49] Karčiauskas K., Peters J. Parameterization transition for guided C2 surfaces of low degree. *Sixth AFA Conference on Curves and Surfaces Avignon* (2006), 183–192.

[50] Karni Z., Gotsman C. Spectral compression of mesh geometry. *Proc. of ACM SIGGRAPH* (2000), 279–286.

[51] Klincsek G. Minimal triangulation of polygonal domains. *Annals of Discrete Mathemathics* **9** (1980), 121–123.

[52] Kruskal J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society* **7** (1956), 48–50.

[53] Langer T., Seidel H. P. *Mean value Bezier surfaces*, Springer (2007), 263–274.

[54] Le B. H., Deng Z. Robust and accurate skeletal rigging from mesh sequences. *SIGGRAPH 2014* **33** (2014), no. 4, p. 84.

[55] Lee C. Y. An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers* (1961), 346–365.

[56] Lee L. P. The nomenclature and classification of map projections. *Empire Survey Review* **7** (1944), 190–200.

[57] Levy B. Laplace-Beltrami eigenfunctions: Towards an algorithm that understands geometry. *IEEE International Conference on Shape Modeling and Applications* (2006), p. 13.

[58] Levy B., Petitjean S., Ray N., Maillot J. Least squares conformal maps for automatic texture atlas generation. *ACM SIGGRAPH* **21** (2002), 362–371.

[59] Liepa P. Filling holes in meshes. *Symposium on Geometry Processing* (2003), 200–205.

[60] Lipman Y., Sorkine O., Cohen-Or D., Levin D., Rössl C., Seidel H.-P. Differential coordinates for interactive mesh editing. *Proc. of Shape Modeling International 04* (2004), 181–190.

[61] Lipnickas A., Raudonis V. Contour representation by clustering curvatures of the 3D objects. *Solid State Phenomena* **147** (2009), 633–638.

[62] Liu L., Zhang L., Xu Y., Gotsman C., Gortler S. J. A local/global approach to mesh parameterization. *Eurographics Symposium on Geometry Processing* **27** (2008), 1495–1504.

[63] Liu L., Zhanga Y., Liub Y., Wangc W. Feature-preserving t-mesh construction using skeleton-based polycubes. *Computer-Aided Design* **58** (2015), 162–172.

[64] Lloyd S. Least square quantization in PCM. *IEEE transactions on information theory* **28** (1982), 129–137.

[65] Loop C., Schaefer S. Approximating Catmull-Clark subdivision surfaces with bicubic patches. *Transactions on Graphics* **27** (2008), no. 1, 8:1–8:11.

[66] Mockus J. A web-based bimatrix game optimization model of polynomial complexity. *Informatica* **20** (2009), 79–98.

[67] Murali T. M., Funkhouser T. A. Consistent solid and boundary representations from arbitrary polygonal data. *Proc. Symposium on Interactive 3D Graphics* (1997), 155–162.

[68] Nealen A., Sorkine O., Alexa M., Cohen-Or D. A sketch-based interface for detail-preserving mesh editing. *Proc. of ACM SIGGRAPH 05* (2005), 1142–1147.

[69] Needham T. *Visual complex analysis*, Oxford Press, 1994.

[70] Nießner M., Loop C. Feature adaptive GPU rendering of Catmull-Clark subdivision surfaces. *ACM Transactions on Graphics* **31** (2012), no. 1, 1–11.

[71] Nooruddin F., Turk G. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics* **9** (2003), no. 2, 191–205.

[72] Ohtake Y., Belyaev A., Bogaevski I. A. Polyhedral surface smoothing with simultaneous mesh regularization. *Proc. of Geometric Modeling and Processing* (2000), 229–237.

[73] Pataky T. C., Mu T., Bosch K., Rosenbaum D., Gait J. Y. Highly unique dynamic plantar pressure patterns among 104 individuals. *J. R. Soc. Interface* **69** (2011), no. 9, 790–800.

[74] Patrikalakis M. P., Maekawa T. Shape interrogation for computer-aided design and manufacturing. *Springer Verlag*, Berlin Heidelberg, 2002.

[75] Pauly M., Mitra N., Giesen J., Gross M., Guibas L. J. Example-based 3D scan completion. *Symposium on Geometry Processing* (2005), 23–32.

[76] Peyré G., Cohen L. Surface segmentation using geodesic centroidal tesselation. *Proceedings 3DPVT'04* (2004), 995–1002.

[77] Peters J., Reift U. Analysis of algorithms generalizing B-spline subdivision. *SIAM Journal on Numerical Analysis* **35** (1996), no. 2, 728–748.

[78] Podolak J., Rusinkiewicz S. Atomic volumes for mesh completion. *Symposium on Geometry Processing* (2005), 33–41.

[79] Polak E. *Optimization: Algorithms and consistent approximations*, Springer-Verlag, 1997.

[80] Reuter M., Wolter F.-E., Peinecke N. Laplace-Beltrami spectra as 'shape-DNA' of surfaces and solids. *Computer-Aided Design* **38** (2006), no. 4, 342–366.

[81] Rodriguez-Quinonez J. C., Sergiyenko O., Tyrsa V., Basaca-Preciado L. C., Rivas-Lopez M., Hernandez-Balbuena D., Pena-Cabrera M. 3D body and medical scanners' technologies: Methodology and spatial discriminations. *Optoelectronic Devices and Properties* **15** (2011), 307–322.

[82] Róth Á., Juhász I. Quadrilateral mesh generation from point clouds by a Monte Carlo method. *Visualization and Computer Vision in co-operation with EUROGRAPHICS* (2009), 97–104.

[83] Sander P., Gortler S., Snyder J., Hoppe H. Signal-specialized parametrization. *Eurographics Association/Association for Computing Machinery* (2002).

[84] Sapidis N. S. *Designing fair curves and surfaces: Shape quality in geometric modeling and computer-aided design*, SIAM, 1994.

[85] Sheffer A., Levy B., Mogilnitsky M., Bogomyakov A. ABF++ : Fast and robust angle based flattening. *ACM Transaction on Graphics* **24** (2005), no. 2, 311–330.

[86] Sheffer A., Sturler E. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers* **73** (2001), no. 3, 326–337.

[87] Shen C., O'Brien J. F., Shewchuk J. R. Interpolating and approximating implicit surfaces from polygon soup. *Proc. of ACM SIGGRAPH 04* (2004), 896–904.

[88] Shewchuk J. R. *Delaunay refinement mesh generation*, PhD thesis, Carnegie Mellon University, Pittsburg (1997).

[89] Shewchuk J. R. What is a good linear element? interpolation, conditioning, and quality measures. *Eleventh International Meshing Roundtable* (2002), 115–126.

[90] Shields R. Cultural topology: The seven bridges of Königsburg, 1736. *Theory Culture Society* **29** (2012), no. 43, 43–57.

[91] Shirangi M. G., Emerickb A. A. An improved TSVD-based Levenberg–Marquardt algorithm for history matching and comparison with Gauss–Newton. *Journal of Petroleum Science and Engineering* **143** (2016), 258–271.

[92] Sorkine O., Cohen-Or D., Lipman Y., Alexa M., Rössl C., Seidel H.-P. Laplacian surface editing. *Proc. of Eurographics symposium on Geometry Processing 04* (2004), 179–188.

[93] Sorkine O., Cohen-Or D., Toledo S. High-pass quantization for mesh encoding. *Symposium on Geometry Processing* (2003), 42–51.

[94] Taubin G. A signal processing approach to fair surface design. *Proc. of ACM SIGGRAPH 95* (1995), 351–358.

[95] Taubin G. *Geometric signal processing on polygonal meshes*, Eurographics 00 State of the Art Report (2000).

[96] Taubin G., Zhang T., Golub G. Optimal surface smoothing as filter design. *European Conference on Computer Vision (ECCV Vol. 1)* (1996), 283–292.

[97] Telfer S., Woodburn J. The use of 3D surface scanning for the measurement and assessment of the human foot. *Journal of Foot and Ankle Research* **3** (2010), 19–28.

[98] Trémaux, C. P. École Polytechnique of Paris (X: 1876). *French engineer of the telegraph in Public conference*, 2010.

[99] Turk G., Levoy M. Zippered polygon meshes from range images. *Proc. of ACM SIGGRAPH 94* (1994), 311–318.

[100] Tutte W. Convex representation of graphs. *Proceedings of the London Mathematical Society* **10** (1960), 304–320.

[101] Valette S., Chassery J.-M. Approximated centroidal Voronoi diagrams for uniform polygonal mesh coarsening. *Computer Graphics Forum (Eurographics proceedings)* **23** (2004), no. 3, 381–389.

[102] Vallet B., Levy B. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum (Proc. Eurographics)* **27** (2008), 251–260.

[103] Wagnera M., Hormannb K., Greiner G. *C2-continuous surface reconstruction with piecewise polynomial patches*, California Institute of Technology, California, USA, 2003.

[104] Waldo T. *Notes and comments on the composition of terrestrial and celestial maps*, ESRI Press, 1972.

[105] Wood Z., Hoppe H., Desbrun M., Schröder P. Removing excess topology from isosurfaces. *ACM Transactions on Graphics* **23** (2004), no. 2, 190–208.

[106] Yifan C., Beier K. P., Papageorgiou D. Direct highlight line modification on nurbs surfaces. *Computer Aided Geometric Design* **14** (1997), no. 6, 583–601.

[107] Yu Y., Zhou K., Xu D., Shi X., Bao H., Guo B., Shum H.-Y. Mesh editing with Poisson-based gradient field manipulation. *Proc. of ACM SIGGRAPH 04* (2004), 644–651.

[108] Zayer R., Rössl C., Karni Z., Seidel H.-P. Harmonic guidance for surface deformation. *Proc. of Eurographics 05* (2005), 601–609.

[109] Zhanguo M., Liu B., Hongbin Z. Skeleton based 3D mesh deformation. *Information, Communications & Signal Processing, 2007* 6th International Conference on (2007), 1–5.

[110] Zhao X., Su Z., Gu X. Area-preservation mapping using optimal mass transport. *IEEE Trans. Vis. Comput. Graph.* **29** (2013), no. 12, 2838–2847.

[111] Zhou K., Huang J., Snyder J., Liu X., Bao H., Guo B., Shum H.-Y. Large mesh deformation using the volumetric graph laplacian. *Proc. of ACM SIGGRAPH 05* (2005), 496–503.

Autoriaus publikacijų sąrašas disertacijos tema

Straipsniai periodiniuose recenzuojamuose leidiniuose

[1A] **Sabaliauskas M.**, Marcinkevičius V. An investigation of ABF++, LSCM, and ARAP methods for parameterization of shoetrees. *Science – Future of Lithuania / Mokslas – Lietuvos Ateitis*, Vilnius, Technika, ISSN 2029-2341, 2015, 7(3): 296–299.

[2A] **Sabaliauskas M.** An algorithm for approximation of Bezier surfaces for special case of quadrangular grid. *Computional Science and Techniques*, Klaipėda, Klaipėda University, ISSN 2029-9966, 2015, 3(2): 454–563.

[3A] **Sabaliauskas M.**, Mockus J. On the Nash equilibrium in the inspector problem. *Lietuvos matematikos rinkinys*, ISSN 0132-2818, 2014, 55: 79–84.

[4A] Mockus J., **Sabaliauskas M.** On the exact polynomial time algorithm for a special class of bimatrix game. *Lietuvos matematikos rinkinys*, ISSN 0132-2818, 2013, 54: 85–90.

Straipsniai recenzuojamuose konferencijų leidiniuose

[5A] **Sabaliauskas M.**, Marcinkevičius V. Segmentation model for flattening of individual 3D lasts. *EMC 2016: 6th International symposium „Engineering Management and Competetiveness“: proceedings*, Kotor, Montenegro, ISBN: 9788676722846, 2016, 325–331.

[6A] **Sabaliauskas M.** A robust SVD algorithm for ARAP method. *EMC 2015: 5th International symposium „Engineering Management and Competetiveness“: proceedings*, Zrejanin, Serbia, University of Novi Sad, ISBN 9788-676722563, 2015, 333–336.

Santraukos konferencijų leidiniuose

[7A] **Sabaliauskas M.**, Marcinkevičius V. An investigation of surface deformation using singular value decomposition. *Data analysis methods for software systems: 8th International Workshop*: [abstracts book], Druskininkai, Lithuania, ISBN 9789986680611, 2016, p. 52.

[8A] **Sabaliauskas M.** Skaitmeninių organinės prigimties kurpalių modelių segmentavimo ir projektavimo į plokštumą tyrimas. *Fizinių ir technologijos mokslų tarpdalykiniai tyrimai: šeštoji jaunųjų mokslininkų konferencija: pranešimų santraukos*, Vilnius, 2016 m. vasario 10 d. Vilnius, Lietuvos mokslų akademijos leidykla, 2016, 57–60.

[9A] **Sabaliauskas M.**, Marcinkevičius V. An Application of ARAP Method for Parameterization of Shoetrees. *Data analysis methods for software systems: 7th International Workshop*: [abstracts book], Druskininkai, Lithuania, ISBN 9789986680581, 2015, 45–46.

[10A] **Sabaliauskas M.**, Marcinkevičius V. Comparison of mods of shoetrees obtained by theoretical and experimental methods. *Data analysis methods for software systems: 6th International Workshop*: [abstracts book], Druskininkai, Lithuania, 2014 [Vilnius], ISBN 9789986680505, 2014, 44.

Trumpos žinios apie autorių

Išsilavinimas:

2012–2016 Vilniaus universitetas (informatikos inžinerijos doktorantūra),

2010–2012 Vilniaus universitetas (matematikos magistras, mokytojas),

2006–2010 Vilniaus universitetas (matematikos bakalauras),

1994–2006 Šiaulių Stasio Šalkauskio vidurinė mokykla (vidurinis išsilavinimas).

Mokslinio darbo patirtis:

2016– Vilniaus universiteto jaunesnysis mokslo darbuotojas,

2016– Vilniaus universiteto specialistas,

2013–2015 Vilniaus universiteto vyriausiasis projekto specialistas.

Priedai (programinės įrangos *Maple* pirminiai programų tekstai)

Paviršiaus išklotinės sudarymo ARAP algoritmas

```
> restart:
> with(MTM):
> with(linalg):

> atan2:=proc(y,x)
> local r;
> if x>0 then r:=arctan(y/x);
> else if x<0 then if y>=0 then r:=arctan(y/x)+Pi; else r:=arctan(y/x)-Pi; end if;
  else if y>0 then r:=Pi/2; else if y<0 then r:=-Pi/2; else r:=undefined; end if;
  end if;  end if; end if;
> return(r);
> end proc;

> SVD22:=proc(a,b,c,d)
> local s11,s22,gamma,beta,u11,u12,u21,u22,v11,v12,v21,v22,palyg1,palyg2,QQQ;
> s11:=1/2*(sqrt((a-d)^2+(b+c)^2)+sqrt((a+d)^2+(b-c)^2));
> s22:=abs(1/2*(sqrt((a+d)^2+(b-c)^2)-sqrt((a-d)^2+(b+c)^2)));
> beta:=1/2*(atan2(-c+b,a+d)-atan2(c+b,-d+a));
> gamma:=1/2*(atan2(c+b,-d+a)+atan2(-c+b,a+d));
> v11:=cos(gamma);
> v12:=sqrt(1-v11^2);
> v21:=v12;
> v22:=v11;
> palyg1:=evalf(s11^2*v11*v12);
> palyg2:=evalf(s22^2*v12*v22);
> if a*b+c*d>0 then
> if palyg1>palyg2 then
> v12:=-v12;
> else
> v11:=-v11;
> end if;
> else
> if palyg1>palyg2 then
> v11:=-v11;
> else
> v12:=-v12;
> end if;
> end if;
> v22:=-v11*v12/v21;
> u11:=(a*v11+b*v21)/s11;
> u12:=(a*v12+b*v22)/s22;
> u21:=(c*v11+d*v21)/s11;
> u22:=(c*v12+d*v22)/s22;QQQ:=evalf(Matrix([[u11,u12],[u21,u22]])),
  evalf(Matrix([[s11,0],[0,s22]])),evalf(Matrix([[v11,v12],[v21,v22]]));
> return(QQQ);
> end proc;

> LoadOBJ:=proc(input)
> local i,j,k,s,m:=1;
> global x,t,vt,xidx:=1,tidx:=1,vtidx:=1;
> fopen(input,READ,TEXT);
> while true do
> s:=fscanf(input, '%s', 1);
> if feof(input) then
> break;
> end if;
> if s[1]="#" then
> s:=fgets(input);
> else if s[1]="v" then
> x[xidx][1]:=fscanf(input, '%f', 1)[];
> x[xidx][2]:=fscanf(input, '%f', 1)[];
> x[xidx][3]:=fscanf(input, '%f', 1)[];
> xidx:=xidx+1;
```

```
> else if s[1]="vt" then
> vt[vtidx][1]:=fscanf(input, '%f', 1)[];
> vt[vtidx][2]:=fscanf(input, '%f', 1)[];
> vtidx:=vtidx+1;
> else if s[1]="f" then
> for k from 1 to 3 do
> t[tidx][k]:=fscanf(input, '%d', 1)[];
> m:=min(m,t[tidx][k]);
> while true do
> s:=fscanf(input, '%c', 1);
> if s[1]="/" then
> fscanf(input, '%d', 1);
> break;
> end if;
> end do;
> end do;
> tidx:=tidx+1;
> end if;
> end if;
> end if;
> end if;
> end do;tidx:=tidx-1;xidx:=xidx-1;vtidx:=vtidx-1;
> if m=0 then
> for i from 1 to tidx do
> for j from 1 to 3 do
> t[i][j]:=t[i][j]+1;
> end do;
> end do;
> end if;
> fclose(input);
> end proc;

> CalEdgeVectorsCalCots:=proc(x,t)
> local i,j,k,v,a,b,c,angle1,angle2,angle3;
> global EV,C,tidx;
> for i from 1 to tidx do
> for j from 1 to 3 do
> for k from 1 to 3 do
> v[k][j]:=x[t[i][k]][j];
> end do;
> end do;
> a:=sqrt((v[1][1]-v[2][1])^2+(v[1][2]-v[2][2])^2+(v[1][3]-v[2][3])^2);
> b:=sqrt((v[2][1]-v[3][1])^2+(v[2][2]-v[3][2])^2+(v[2][3]-v[3][3])^2);
> c:=sqrt((v[3][1]-v[1][1])^2+(v[3][2]-v[1][2])^2+(v[3][3]-v[1][3])^2);
> angle1:=arccos((a^2+c^2-b^2)/(2*a*c));
> angle2:=arccos((a^2+b^2-c^2)/(2*a*b));
> angle3:=arccos((b^2+c^2-a^2)/(2*b*c));
> C[i][1]:=cot(angle1);
> C[i][2]:=cot(angle2);
> C[i][3]:=cot(angle3);
> EV[i][1][1]:=c*cos(angle1)-a;
> EV[i][2][1]:=-c*cos(angle1);
> EV[i][3][1]:=a;
> EV[i][1][2]:=c*sin(angle1);
> EV[i][2][2]:=-c*sin(angle1);
> EV[i][3][2]:=0;
> end do;
> return();
> end proc;

> laplacian2:=proc(x,t,C)
> local i,j,j1,j2,j3,idx1,idx2,idx3;
> global L,tidx,xidx;
> L:=Matrix(xidx+1,xidx+1, storage=sparse);
> for i from 1 to tidx do
> for j from 0 to 2 do
> j1:=j+1;
> j2:=`mod`(j+1,3)+1;
> j3:=`mod`(j+2,3)+1;
> idx1:=t[i][j1];
> idx2:=t[i][j2];
> idx3:=t[i][j3];
> L[idx1,idx1]:=L[idx1,idx1]+C[i][j2]+C[i][j3];
> L[idx1,idx2]:=L[idx1,idx2]-C[i][j3];
```

```
> L[idx1,idx3]:=L[idx1,idx3]-C[i][j2];
> end do;
> end do;
> return();
> end proc;

> ARAP_Local:=proc(u,t,EV,C)
> local i,j,k,idx1,idx2,idx3,u1,u2,u3,CovMat,s,v,d,rot,uu,xx,cc;
> global R,tidx;
> for i from 1 to tidx do
> idx1:=t[i][1];
> idx2:=t[i][2];
> idx3:=t[i][3];
> for j from 1 to 2 do
> u1[j]:=vt[idx1][j];
> u2[j]:=vt[idx2][j];
> u3[j]:=vt[idx3][j];
> uu[1][j]:=u3[j]-u2[j];
> uu[2][j]:=u1[j]-u3[j];
> uu[3][j]:=u2[j]-u1[j];
> end do;
> xx[1][1]:=EV[i][1][1];
> xx[1][2]:=EV[i][1][2];
> xx[2][1]:=EV[i][2][1];
> xx[2][2]:=EV[i][2][2];
> xx[3][1]:=EV[i][3][1];
> xx[3][2]:=EV[i][3][2];
> cc[1][1]:=C[i][1];
> cc[2][2]:=C[i][2];
> cc[3][3]:=C[i][3];
> CovMat[1][1]:=cc[1][1]*uu[1][1]*xx[1][1]+cc[2][2]*uu[2][1]*xx[2][1]+
  cc[3][3]*uu[3][1]*xx[3][1];
> CovMat[1][2]:=cc[1][1]*uu[1][2]*xx[1][1]+cc[2][2]*uu[2][2]*xx[2][1]+
  cc[3][3]*uu[3][2]*xx[3][1];
> CovMat[2][1]:=cc[1][1]*uu[1][1]*xx[1][2]+cc[2][2]*uu[2][1]*xx[2][2]+
  cc[3][3]*uu[3][1]*xx[3][2];
> CovMat[2][2]:=cc[1][1]*uu[1][2]*xx[1][2]+cc[2][2]*uu[2][2]*xx[2][2]+
  cc[3][3]*uu[3][2]*xx[3][2];
> s,v,d:=SVD22(CovMat[1][1],CovMat[1][2],CovMat[2][1],CovMat[2][2]);
> rot[1][1]:=d[1][1]*s[1][1]+d[1][2]*s[1][2];
> rot[1][2]:=d[1][1]*s[2][1]+d[1][2]*s[2][2];
> rot[2][1]:=d[2][1]*s[1][1]+d[2][2]*s[1][2];
> rot[2][2]:=d[2][1]*s[2][1]+d[2][2]*s[2][2];
> if rot[1][1]*rot[2][2]-rot[1][2]*rot[2][1]<0 then
> if v[1][1]<v[2][2] then
> rot[1][1]:=-d[1][1]*s[1][1]+d[1][2]*s[1][2];
> rot[1][2]:=-d[1][1]*s[2][1]+d[1][2]*s[2][2];
> rot[2][1]:=-d[2][1]*s[1][1]+d[2][2]*s[1][2];
> rot[2][2]:=-d[2][1]*s[2][1]+d[2][2]*s[2][2];
> else
> rot[1][1]:=d[1][1]*s[1][1]-d[1][2]*s[1][2];
> rot[1][2]:=d[1][1]*s[2][1]-d[1][2]*s[2][2];
> rot[2][1]:=d[2][1]*s[1][1]-d[2][2]*s[1][2];
> rot[2][2]:=d[2][1]*s[2][1]-d[2][2]*s[2][2];
> end if;
> end if;
> R[i][1]:=rot[1][1];
> R[i][2]:=rot[1][2];
> R[i][3]:=rot[2][1];
> R[i][4]:=rot[2][2];
> end do;
> return();
> end proc;

> ARAP_Global:=proc(EV,Linv,t,C,R)
> local i,j,k,Bx,By,j1,j2,j3,idx2,idx3,Ux,Uy,sum1,sum2,Eij,rot;
> global U,vt,xidx,tidx;
> for i from 1 to xidx+1 do
> Bx[i]:=0;
> By[i]:=0;
> end do;
> for i from 1 to tidx do
> for j from 0 to 2 do
> j1:=j+1;
```

```
> j2:=`mod`(j+1,3)+1;
> j3:=`mod`(j+2,3)+1;
> idx2:=t[i][j2];
> idx3:=t[i][j3];
> Eij[1][1]:=EV[i][j1][1];
> Eij[2][1]:=EV[i][j1][2];
> rot[1][1]:=R[i][1];
> rot[1][2]:=R[i][2];
> rot[2][1]:=R[i][3];
> rot[2][2]:=R[i][4];
> for k from 1 to 2 do
> Bx[idx3]:=Bx[idx3]+rot[1][k]*Eij[k][1]*C[i][j1];
> Bx[idx2]:=Bx[idx2]-rot[1][k]*Eij[k][1]*C[i][j1];
> By[idx3]:=By[idx3]+rot[2][k]*Eij[k][1]*C[i][j1];
> By[idx2]:=By[idx2]-rot[2][k]*Eij[k][1]*C[i][j1];
> end do;
> end do;
> end do;
> Bx[xidx+1]:=0;
> By[xidx+1]:=0;
> sum1:=0;
> sum2:=0;
> for i from 1 to xidx+1 do
> for j from 1 to xidx+1 do
> sum1:=sum1+Linv[i,j]*Bx[j];
> sum2:=sum2+Linv[i,j]*By[j];
> end do;
> Ux[i]:=sum1;
> Uy[i]:=sum2;
> sum1:=0;
> sum2:=0;
> end do;
> for i from 1 to xidx do
> U[i][1]:=Ux[i];vt[i][1]:=U[i][1];
> U[i][2]:=Uy[i];vt[i][2]:=U[i][2];
> end do;
> return();
> end proc;

> CalRigidEnergy:=proc(EV,t,U,C,R)
> local i,j,k,Uij,Eij,rot,j1,j2,j3,idx2,idx3;
> global E:=0;
> for i from 1 to tidx do
> for j from 0 to 2 do
> j1:=j+1;
> j2:=`mod`(j+1,3)+1;
> j3:=`mod`(j+2,3)+1;
> idx2:=t[i][j2];
> idx3:=t[i][j3];
> for k from 1 to 2 do
> Uij[k]:=U[idx3][k]-U[idx2][k];
> end do;
> Eij[1][1]:=EV[i][j1][1];
> Eij[2][1]:=EV[i][j1][2];
> rot[1][1]:=R[i][1];
> rot[1][2]:=R[i][2];
> rot[2][1]:=R[i][3];
> rot[2][2]:=R[i][4];
> E:=E+C[i][j1]*((Uij[1]-Eij[1][1]*rot[1][1]-Eij[2][1]*rot[1][2])^2+
  (Uij[2]-Eij[1][1]*rot[2][1]-Eij[2][1]*rot[2][2])^2);
> end do;
> end do;
> return();
> end proc;

> WriteOBJ:=proc(output,x,t,vt)
> local i,j,k;
> global xidx,tidx,vtidx;
> fopen(output,WRITE,TEXT);
> fprintf(output, "%s\n%s\n%s\n\n","#","# Wavefront OBJ","#");
> fprintf(output, "# %d vertices, %d triangles, 1 groups\n\n",xidx,tidx);
> fprintf(output, "vn 0 0 1\n");
> for i from 1 to xidx do
> fprintf(output, "v %f %f %f\n", x[i][1], x[i][2], x[i][3]);
```

```
> end do;
> fprintf(output, "\n");
> for i from 1 to vtidx do
> fprintf(output, "vt %f %f\n", vt[i][1], vt[i][2]);
> end do;
> fprintf(output, "\n");
> for i from 1 to tidx do
> fprintf(output, "f %d/%d %d/%d %d/%d\n", t[i][1],t[i][1],t[i][2],t[i][2],
  t[i][3],t[i][3]);
> end do;
> fclose(output);
> return();
> end proc;

> ARAP:=proc(input,output)
> local i,j,k,Epre,iterations,E,Linv;
> global C,R,L,EV;
> LoadOBJ(input);
> CalEdgeVectorsCalCots(x,t);
> laplacian2(x,t,C);
> L[1,xidx+1]:=1;
> L[xidx+1,1]:=1;
> Linv:=inverse(Matrix([seq([seq(L[i,j],j=1..xidx+1)],i=1..xidx+1)]));
> E:=-1;
> Epre:=0;
> iterations:=0;
> while abs(Epre-E)>0.001 do
> iterations:=iterations+1;
> Epre:=E;
> ARAP_Local(vt,t,EV,C);
> ARAP_Global(EV,Linv,t,C,R);
> CalRigidEnergy(EV,t,vt,C,R);
> end do;
> WriteOBJ(output,x,t,vt);
> return();
> end proc;

> input:=`C:/Users/akatasis/Desktop/ARAP-matlab/test123.obj`:
> output:=`C:/Users/akatasis/Desktop/ARAP-matlab/rez126.obj`:
> ARAP(input,output);
```

Skaitmeninio kurpalio modelio paviršiaus segmentavimo algoritmas

```
> restart;

> LoadOBJ:=proc(input)
> local i,j,k,s,m:=1;
> global x,t,vt,xidx:=1,tidx:=1,vtidx:=1;
> fopen(input,READ,TEXT);
> while true do
> s:=fscanf(input, '%s', 1);
> if feof(input) then
> break;
> end if;
> if s[1]="#" then
> s:=fgets(input);
> else if s[1]="v" then
> x[xidx][1]:=fscanf(input, '%f', 1)[];
> x[xidx][2]:=fscanf(input, '%f', 1)[];
> x[xidx][3]:=fscanf(input, '%f', 1)[];
> xidx:=xidx+1;
> else if s[1]="vt" then
> vt[vtidx][1]:=fscanf(input, '%f', 1)[];
> vt[vtidx][2]:=fscanf(input, '%f', 1)[];
> vtidx:=vtidx+1;
> else if s[1]="f" then
> for k from 1 to 3 do
> t[tidx][k]:=fscanf(input, '%d', 1)[];
> m:=min(m,t[tidx][k]);
> while true do
> s:=fscanf(input, '%c', 1);
> if s[1]="/" then
> fscanf(input, '%d', 1);
> break;
> end if;
> end do;
> end do;
> tidx:=tidx+1;
> end if;
> end if;
> end if;
> end if;
> end do;tidx:=tidx-1;xidx:=xidx-1;vtidx:=vtidx-1;
> if m=0 then
> for i from 1 to tidx do
> for j from 1 to 3 do
> t[i][j]:=t[i][j]+1;
> end do;
> end do;
> end if;
> fclose(input);
> end proc;

> normales:=proc()
> local i,j,k,nrm,a,b;
> global kamp,nbh;
> for i from 1 to xidx do
> nbh[i]:={};
> nrm[i]:=[0,0,0];
> end do;
> for j from 1 to tidx do
> for k from 1 to 3 do
> nbh[t[j][k]]:=`minus`(`union`(nbh[t[j][k]],t[j][1],t[j][2],t[j][3]),t[j][k]);
> a[k]:=x[t[j][2]][k]-x[t[j][3]][k];
> b[k]:=x[t[j][3]][k]-x[t[j][1]][k];
> end do;
> nrm[j]:=[a[2]*b[3]-a[3]*b[2],a[3]*b[1]-a[1]*b[3],a[1]*b[2]-a[2]*b[1]];
> end do;
> for i from 1 to tidx do
> kamp[i]:=arcsin(nrm[i][3]/sqrt(nrm[i][1]^2+nrm[i][2]^2+nrm[i][3]^2));
> end do;
> return();
```

```
> end proc;

> areas:=proc(lower_angle,upper_angle)
> local i,j,srit,mid_z,n,top_start,bottom_start;
> global kont,top_start_x,bottom_start_x;
> n:=0;
> mid_z:=0;
> for i from 1 to xidx do
> mid_z:=mid_z+x[i][3];
> n:=n+1;
> for j from 1 to 3 do
> srit[i][j]:=false;
> end do;
> end do;
> top_start:=mid_z/n;
> bottom_start:=top_start;
> for i from 1 to tidx do
> if kamp[i]<lower_angle then
> for j from 1 to 3 do
> srit[t[i][j]][1]:=true
> end do;
> else if kamp[i]>upper_angle then
> for j from 1 to 3 do
> srit[t[i][j]][2]:=true
> end do;
> else
> for j from 1 to 3 do
> srit[t[i][j]][3]:=true
> end do;
> end if;
> end if;
> end do;
> for i from 1 to xidx do
> if srit[i][3]=true then
> if srit[i][1]=true or srit[i][2]=true then
> kont[i]:=true;
> if x[i][3]>top_start then
> top_start_x:=i;
> top_start:=x[i][3];
> else if x[i][3]<bottom_start then
> bottom_start_x:=i;
> bottom_start:=x[i][3];
> end if;
> end if;
> else kont[i]:=false;
> end if;
> end if;
> end do;
> return();
> end proc;

> konturas:=proc(st)
> local i,opt,q;
> global nbh,kntr,visited,k;
> kntr:='kntr';
> for i from 1 to xidx do
> visited[i]:=false;
> end do;
> q:=p_plot(st);
> for i in nbh[q] do
> if kont[i]=true then
> opt:=i;
> end if;
> end do;
> kntr[1]:=q;
> kntr[2]:=opt;
> visited[opt]:=true;
> k:=3;
> nbh[opt]:=`minus`(nbh[opt],q);
> kirpimas(q,opt,q);
> nbh[opt]:=`union`(nbh[opt],q);
> kntr:=tempimas(kntr,k-1); kntr:=tempimas(kntr,nops(kntr));
> kntr:=tempimas(kntr,nops(kntr));kntr:=tempimas(kntr,nops(kntr));
> return(kntr);
```

```
> end proc;

> kirpimas:=proc(p,q,end_point)
> local i,opt,vekt2;
> global nbh,kamp,kntr,visited,k;
> opt:=0;
> for i in nbh[q] do
> if kont[i]=true and visited[i]=false then
> opt:=i;
> end if;
> end do;
> if opt<>end_point then
> if opt<>0 then
> visited[opt]:=true;
> kntr[k]:=opt;
> k:=k+1;
> kirpimas(q,opt,end_point);
> else
> k:=k-1;
> kirpimas(kntr[k-2],kntr[k-1],end_point);
> end if;
> end if;
> return();
> end proc;

> visit:=proc(q)
> local i,sk;
> global p_plot_visited,ats;
> p_plot_visited[q]:=true;
> sk:=0;
> for i in nbh[q] do
> if kont[i]=true then
> sk:=sk+1;
> end if;
> end do;
> if sk<>2 then
> for i in nbh[q] do
> if kont[i]=true and p_plot_visited[i]=false then
> visit(i);
> end if;
> end do;
> else ats:=q;
> end if;
> end proc;

> p_plot:=proc(q)
> local i;
> global p_plot_visited,ats;
> for i from 1 to xidx do
> p_plot_visited[i]:=false;
> end do;
> visit(q);
> return(ats);
> end proc;

> tempimas:=proc(kont,k)
> local i,j,kont2;
> j:=2;
> kont2[1]:=kont[1];
> for i from 1 to k-2 do
> if is(kont[i] in nbh[kont[i+2]]) then
> kont2[j]:=kont[i+2];
> j:=j+1;
> i:=i+1;
> else
> kont2[j]:=kont[i+1];
> j:=j+1;
> end if;
> end do;
> if kont2[j-1]<>kont[k] then
> kont2[j]:=kont[k]
> end if;
> return(convert(kont2,list));
> end proc;
```

```
> remesh_konturas:=proc(konturas)
> local i,j,laik,temp_remesh;
> global x,k;
> laik[1]:=konturas[nops(konturas)];
> for i from 1 to nops(konturas) do
> laik[i+1]:=konturas[i];
> end do;
> laik[nops(konturas)+2]:=konturas[1];
> for i from 1 to nops(konturas) do
> for j from 1 to 3 do
> temp_remesh[i][j]:=1/4*x[laik[i]][j]+1/2*x[laik[i+1]][j]+1/4*x[laik[i+2]][j];
> end do;
> end do;
> for i from 1 to nops(konturas) do
> for j from 1 to 3 do
> x[konturas[i]][j]:=temp_remesh[i][j];
> end do;
> end do;
> return();
> end proc;

> apat_taskai:=proc(konturas,konturas2,t)
> local i,j,dist,dist2,A,B,kart1,T1,kart2,T2,kart3,T3,seeds;
> global p;
> dist:=0;
> for i from 1 to nops(konturas) do
> for j from i+1 to nops(konturas) do
> dist2:=sqrt((x[konturas[j]][1]-x[konturas[i]][1])^2+(x[konturas[j]][2]-
  x[konturas[i]][2])^2+(x[konturas[j]][3]-x[konturas[i]][3])^2);
> if dist2>dist then
> dist:=dist2;
> A:=konturas[i];
> B:=konturas[j];
> end if;
> end do;
> end do;
> kart1:=0;
> kart2:=0;
> kart3:=0;
> T1:=[0,0,0];
> T2:=[0,0,0];
> T3:=[0,0,0];
> for i from 1 to nops(konturas) do
> if (x[konturas[i]][1]-(1-t)*x[A][1]-x[B][1]*t)*(x[A][1]-x[B][1])+
  (x[konturas[i]][2]-(1-t)*x[A][2]-x[B][2]*t)*(x[A][2]-x[B][2])+
  (x[konturas[i]][2]-(1-t)*x[A][3]-x[B][3]*t)*(x[A][3]-x[B][3])>0 then
> for j from 1 to 3 do
> T1[j]:=T1[j]+x[konturas[i]][j];
> end do;
> kart1:=kart1+1;
> else if (x[konturas[i]][1]-t*x[A][1]-x[B][1]*(1-t))*(x[A][1]-x[B][1])+
  (x[konturas[i]][2]-t*x[A][2]-x[B][2]*(1-t))*(x[A][2]-x[B][2])+
  (x[konturas[i]][2]-t*x[A][3]-x[B][3]*(1-t))*(x[A][3]-x[B][3])<0 then
> for j from 1 to 3 do
> T2[j]:=T2[j]+x[konturas[i]][j];
> end do;
> kart2:=kart2+1;
> end if;
> end if;
> end do;
> for i from 1 to nops(konturas2) do
> for j from 1 to 3 do
> T3[j]:=T3[j]+x[konturas2[i]][j];
> end do;
> kart3:=kart3+1;
> end do;
> T1:=T1/kart1;
> T2:=T2/kart2;
> T3:=T3/kart3;
> p[1]:=T1[2]*T2[3]-T1[2]*T3[3]-T2[2]*T1[3]+T2[2]*T3[3]+T3[2]*T1[3]-T3[2]*T2[3];
> p[2]:=-T1[1]*T2[3]+T1[1]*T3[3]+T2[1]*T1[3]-T2[1]*T3[3]-T3[1]*T1[3]+T3[1]*T2[3];
> p[3]:=T1[1]*T2[2]-T1[1]*T3[2]-T2[1]*T1[2]+T2[1]*T3[2]+T3[1]*T1[2]-T3[1]*T2[2];
> p[4]:=-p[1]*T1[1]-p[2]*T1[2]-p[3]*T1[3];
```

```
> seeds:=seed_points(T2,T3);
> return([seeds]);
> end proc;

> seed_points:=proc(T2,T3)
> local i,dist1,dist2,laik1,laik2,opt1,opt2;
> opt1:=1;
> opt2:=1;
> dist1:=sqrt((T2[1]-x[1][1])^2+(T2[2]-x[1][2])^2+(T2[3]-x[1][3])^2);
> dist2:=sqrt((T3[1]-x[1][1])^2+(T3[2]-x[1][2])^2+(T3[3]-x[1][3])^2);
> for i from 2 to xidx do
> laik1:=sqrt((T2[1]-x[i][1])^2+(T2[2]-x[i][2])^2+(T2[3]-x[i][3])^2);
> laik2:=sqrt((T3[1]-x[i][1])^2+(T3[2]-x[i][2])^2+(T3[3]-x[i][3])^2);
> if laik1<dist1 then
> dist1:=laik1;
> opt1:=i;
> end if;
> if laik2<dist2 then
> dist2:=laik2;
> opt2:=i;
> end if;
> end do;
> return(opt1,opt2);
> end proc;

> atstumas:=proc(p,x,y,z)
> return(abs(p[1]*x+p[2]*y+p[3]*z+p[4])/sqrt(p[1]^2+p[2]^2+p[3]^2));
> end proc;

> seed_x:=proc(x_kont)
> local a,b,dist1,dist2,i,j,temp_kont,ats;
> temp_kont[1]:=x_kont[nops(x_kont)];
> for i from 1 to nops(x_kont) do
> temp_kont[i+1]:=x_kont[i];
> end do;
> j:=1;
> for i from 1 to nops(x_kont) do
> a:=p[1]*x[temp_kont[i]][1]+p[2]*x[temp_kont[i]][2]+p[3]*x[temp_kont[i]][3]+p[4];
> b:=p[1]*x[temp_kont[i+1]][1]+p[2]*x[temp_kont[i+1]][2]+p[3]*x[temp_kont[i+1]][3]+p[4];
> if a*b<=0 then
> dist1:=atstumas(p,x[temp_kont[i]][1],x[temp_kont[i]][2],x[temp_kont[i]][3]);
> dist2:=atstumas(p,x[temp_kont[i+1]][1],x[temp_kont[i+1]][2],x[temp_kont[i+1]][3]);
> if dist1<dist2 then
> ats[j]:=temp_kont[i];
> j:=j+1;
> else
> ats[j]:=temp_kont[i+1];
> j:=j+1;
> end if;
> end if;
> end do;
> return(convert(ats,list));
> end proc;

> seed_plane:=proc(a_kont,v_kont)
> local a,b,dist1,dist2,dist3,dist4;
> a:=seed_x(a_kont);
> b:=seed_x(v_kont);
> dist1:=sqrt((x[b[1]][1]-x[a[1]][1])^2+(x[b[1]][2]-x[a[1]][2])^2+(x[b[1]][3]
  -x[a[1]][3])^2);
> dist2:=sqrt((x[b[1]][1]-x[a[2]][1])^2+(x[b[1]][2]-x[a[2]][2])^2+(x[b[1]][3]
  -x[a[2]][3])^2);
> dist3:=sqrt((x[b[2]][1]-x[a[1]][1])^2+(x[b[2]][2]-x[a[1]][2])^2+(x[b[2]][3]
  -x[a[1]][3])^2);
> dist4:=sqrt((x[b[2]][1]-x[a[2]][1])^2+(x[b[2]][2]-x[a[2]][2])^2+(x[b[2]][3]
  -x[a[2]][3])^2);
> if max(dist1,dist2,dist3,dist4)=dist2 or max(dist1,dist2,dist3,dist4)=dist3 then
> return([[b[1],a[1]],[b[2],a[2]]]);
> else
> return([[b[1],a[2]],[b[2],a[1]]]);
> end if;
> end proc;

> projekt:=proc(s,p)
```

```
> local i,t;
> global x;
> t:=(p[1]*x[s][1]+p[2]*x[s][2]+p[3]*x[s][3]+p[4])/(p[1]^2+p[2]^2+p[3]^2);
> for i from 1 to 3 do
> x[s][i]:=x[s][i]-p[i]*t;
> end do;
> return();
> end proc;

> projekcija:=proc(pr,pab)
> local i,dist,dist2,opt,vekt,vekt2;
> global p,k,atkarpa;
> atkarpa:='atkarpa';
> projekt(pr,p);
> atkarpa[1]:=pr;
> vekt2:=[x[pab][1]-x[pr][1],x[pab][2]-x[pr][2],x[pab][3]-x[pr][3]];
> dist:=infinity;
> for i in nbh[pr] do
> vekt:=[x[i][1]-x[pr][1],x[i][2]-x[pr][2],x[i][3]-x[pr][3]];
> if arccos((vekt[1]*vekt2[1]+vekt[2]*vekt2[2]+vekt[3]*vekt2[3])/sqrt((vekt[1]^2+
  vekt[2]^2+vekt[3]^2)*(vekt2[1]^2+vekt2[2]^2+vekt2[3]^2)))<evalf(Pi/2) then
> dist2:=atstumas(p,x[i][1],x[i][2],x[i][3]);
> if dist2<dist then
> dist:=dist2;
> opt:=i;
> end if;
> end if;
> end do;
> atkarpa[2]:=opt;
> k:=3;
> vekt:=[x[opt][1]-x[pr][1],x[opt][2]-x[pr][2],x[opt][3]-x[pr][3]];
> kirpimas2(opt,vekt,pab);
> return(convert(atkarpa,list));
> end proc;

> kirpimas2:=proc(q,vekt,pab)
> local i,dist,dist2,opt,vekt2;
> global p,atkarpa,k;
> dist:=infinity;
> for i in nbh[q] do
> vekt2:=[x[i][1]-x[q][1],x[i][2]-x[q][2],x[i][3]-x[q][3]];
> if arccos((vekt[1]*vekt2[1]+vekt[2]*vekt2[2]+vekt[3]*vekt2[3])/sqrt((vekt[1]^2+
  vekt[2]^2+vekt[3]^2)*(vekt2[1]^2+vekt2[2]^2+vekt2[3]^2)))<evalf(Pi/2) then
> dist2:=atstumas(p,x[i][1],x[i][2],x[i][3]);
> if dist2<dist then
> dist:=dist2;
> opt:=i;
> end if;
> end if;
> end do;
> if opt<>pab then
> vekt2:=[x[opt][1]-x[q][1],x[opt][2]-x[q][2],x[opt][3]-x[q][3]];
> atkarpa[k]:=opt;
> k:=k+1;
> kirpimas2(opt,vekt2,pab);
> else
> atkarpa[k]:=pab;
> end if;
> return();
> end proc;

> projektavimas2:=proc(proj1,proj2)
> local i,j;
> for i from 1 to nops(proj1) do
> projekt(proj1[i],p);
> end do;
> for j from 1 to nops(proj2) do
> projekt(proj2[j],p);
> end do;
> return();
> end proc;

> remesh_pjuvis:=proc(atkarpa)
> local i,j,laik,temp_remesh;
```

```
> global x,k;
> for i from 1 to nops(atkarpa) do
> laik[i]:=atkarpa[i];
> end do;
> for i from 2 to nops(atkarpa)-1 do
> for j from 1 to 3 do
> temp_remesh[i][j]:=1/4*x[laik[i-1]][j]+1/2*x[laik[i]][j]+1/4*x[laik[i+1]][j];
> end do;
> end do;
> for i from 2 to nops(atkarpa)-1 do
> for j from 1 to 3 do
> x[atkarpa[i]][j]:=temp_remesh[i][j];
> end do;
> end do;
> return();
> end proc;

> barjeras:=proc(kont1,kont2)
> local i,j,k;
> global praeinamas;
> for i from 1 to xidx do
> praeinamas[i]:=true;
> end do;
> for j in kont1 do
> praeinamas[j]:=false;
> end do;
> for k in kont2 do
> praeinamas[k]:=false;
> end do;
> return();
> end proc;

> p_plot_seeds:=proc(s1,kontur1,s2,kontur2)
> local i;
> global p_plot_visited_seeds,area,praeinamas;
> barjeras(kontur1,kontur2);
> for i from 1 to xidx do
> p_plot_visited_seeds[i]:=false;
> area[i]:=1;
> end do;
> p_plot_visited_seeds[s1]:=true;
> area[s1]:=3;
> visit_area(s1,kontur1,3);
> p_plot_visited_seeds[s2]:=true;
> area[s2]:=4;
> visit_area(s2,kontur2,4);
> return();
> end proc;

> visit_area:=proc(q,kontur,srit)
> local i;
> global p_plot_visited_seeds,area,praeinamas;
> p_plot_visited_seeds[q]:=true;
> area[q]:=srit;
> for i in nbh[q] do
> if praeinamas[i]=false then
> area[i]:=srit;
> else if p_plot_visited_seeds[i]=false then
> visit_area(i,kontur,srit);
> end if;
> end if;
> end do;
> end proc;

> segmentai:=proc(p)
> local i;
> global area,segm;
> for i from 1 to tidx do
> if area[t[i][1]]=3 and area[t[i][2]]=3 and area[t[i][3]]=3 then
> segm[i]:=3;
> else if area[t[i][1]]=4 and area[t[i][2]]=4 and area[t[i][3]]=4 then
> segm[i]:=4;
> else if p[1]*(x[t[i][1]][1]+x[t[i][2]][1]+x[t[i][3]][1])/3+p[2]*(x[t[i][1]][2]+
  x[t[i][2]][2]+x[t[i][3]][2])/3+p[3]*(x[t[i][1]][3]+x[t[i][2]][3]+x[t[i][3]][3])/3+p[4]>0
```

```
> then segm[i]:=1;
> else segm[i]:=2;
> end if;
> end if;
> end if;
> end do;
> return();
> end proc;

> makeoff:=proc(output)
> local i,j;
> fopen(output,WRITE,TEXT):
> fprintf(output,"%s\n%s\n%d %d %d\n","# created by akatasis","OFF",xidx,tidx,xidx+tidx-2);
> for i from 1 to xidx do
> fprintf(output,"%f %f %f\n",x[i][1],x[i][2],x[i][3]);
> end do;
> for j from 1 to tidx do
> fprintf(output,"%d %d %d %d %d %d %d\n",3,t[j][1]-1,t[j][2]-1,t[j][3]-1,
  spalva[segm[j]][1],spalva[segm[j]][2],spalva[segm[j]][3]);
> end do;
> fclose(output):
> return();
> end proc;

> makeoff_part:=proc(output2,sritis)
> local i,j,k,sk,aibe,taskai,eile,transf_t,last;
> sk:=0;
> for i from 1 to tidx do
> if segm[i]=sritis then
> sk:=sk+1;
> transf_t[sk]:=i;
> taskai[3*sk-2]:=t[i][1];
> taskai[3*sk-1]:=t[i][2];
> taskai[3*sk]:=t[i][3];
> end if;
> end do;
> taskai:=convert(taskai,set);
> k:=1;last:=taskai[nops(taskai)];
> for i from 1 to last do
> if i=taskai[k] then
> eile[i]:=k;
> k:=k+1;
> end if;
> end do;
> fopen(output2,WRITE,TEXT):
> fprintf(output2,"%s\n%s\n%d %d %d\n","# created by akatasis","OFF",nops(taskai),
  sk,nops(taskai)+sk-2);
> for i from 1 to nops(taskai) do
> fprintf(output2,"%f %f %f\n",x[taskai[i]][1],x[taskai[i]][2],x[taskai[i]][3]);
> end do;
> for j from 1 to sk do
> fprintf(output2,"%d %d %d %d %d %d %d\n",3,eile[t[transf_t[j]][1]]-1,
  eile[t[transf_t[j]][2]]-1,eile[t[transf_t[j]][3]]-1,spalva[sritis][1],
  spalva[sritis][2],spalva[sritis][3]);
> end do;
> fclose(output2):
> return();
> end proc;

> makeobj_part:=proc(output2,sritis)
> local i,j,k,sk,aibe,taskai,eile,transf_t,last;
> sk:=0;
> for i from 1 to tidx do
> if segm[i]=sritis then
> sk:=sk+1;
> transf_t[sk]:=i;
> taskai[3*sk-2]:=t[i][1];
> taskai[3*sk-1]:=t[i][2];
> taskai[3*sk]:=t[i][3];
> end if;
> end do;
> taskai:=convert(taskai,set);
> k:=1;last:=taskai[nops(taskai)];
> for i from 1 to last do
```

```
> if i=taskai[k] then
> eile[i]:=k;
> k:=k+1;
> end if;
> end do;
> fopen(output2,WRITE,TEXT):
> for i from 1 to nops(taskai) do
> fprintf(output2,"%s %f %f %f\n","v", x[taskai[i]][1],x[taskai[i]][2],x[taskai[i]][3]);
> end do;
> for i from 1 to nops(taskai) do
> fprintf(output2,"%s %f %f\n","vt", 0, 0);
> end do;
> for j from 1 to sk do
> fprintf(output2,"%s %d%s%d %d%s%d %d%s%d\n","f", eile[t[transf_t[j]][1]],"/",
  eile[t[transf_t[j]][1]],eile[t[transf_t[j]][2]],"/",eile[t[transf_t[j]][2]],
  eile[t[transf_t[j]][3]],"/",eile[t[transf_t[j]][3]]);
> end do;
> fclose(output2):
> return();
> end proc;

> kurpalis_remesh:=proc()
> local i,j,k,temp_mesh,koef;
> global x;
> for i from 1 to xidx do
> for j from 1 to 3 do
> temp_mesh[i][j]:=0;
> end do;
> koef:=2*nops(nbh[i]);
> for k in nbh[i] do
> for j from 1 to 3 do
> temp_mesh[i][j]:=temp_mesh[i][j]+x[k][j];
> end do;
> end do;
> for j from 1 to 3 do
> temp_mesh[i][j]:=x[i][j]/2+temp_mesh[i][j]/koef;
> end do;
> end do;
> for i from 1 to xidx do
> for j from 1 to 3 do
> x[i][j]:=temp_mesh[i][j];
> end do;
> end do;
> return();
> end proc;

> galutinis:=proc(input,output)
> global apat_kont,virs_kont,ss,sp,pr1,pr2,spalva;
> LoadOBJ(input);
> normales();kurpalis_remesh();kurpalis_remesh();kurpalis_remesh();kurpalis_remesh();
> areas(evalf(-Pi/4),evalf(Pi/4));
> apat_kont:=konturas(bottom_start_x):
> virs_kont:=konturas(top_start_x):
> remesh_konturas(apat_kont);
> remesh_konturas(apat_kont);
> remesh_konturas(apat_kont);
> remesh_konturas(apat_kont);
> remesh_konturas(apat_kont);
> remesh_konturas(virs_kont);
> remesh_konturas(virs_kont);
> ss:=apat_taskai(apat_kont,virs_kont,1/3);
> sp:=seed_plane(apat_kont,virs_kont);
> pr1:=projekcija(sp[1][1],sp[1][2]):
> pr2:=projekcija(sp[2][1],sp[2][2]):
> pr1:=tempimas(pr1,nops(pr1));
> pr2:tempimas(pr2,nops(pr2));
> projektavimas2(pr1,pr2);
> remesh_pjuvis(pr1);
> remesh_pjuvis(pr2);
> p_plot_seeds(ss[1],apat_kont,ss[2],virs_kont);
> segmentai(p);
> spalva:=[[255,0,0],[0,255,0],[0,0,255],[255,255,0]];
> makeoff(output,spalva);
> end proc;
```

```
> input:='C:/Users/akatasis/Desktop/segmentavimas/eksperimentai/obj/Right1.obj';
> output:='C:/Users/akatasis/Desktop/segmentavimas/v3/nauji2015/Right1.off';
> output2:='C:/Users/akatasis/Desktop/segmentavimas/v3/nauji2015/Right1_L.obj';
> output3:='C:/Users/akatasis/Desktop/segmentavimas/v3/nauji2015/Right1_R.obj';
> output4:='C:/Users/akatasis/Desktop/segmentavimas/v3/nauji2015/Right1_L.off';
> output5:='C:/Users/akatasis/Desktop/segmentavimas/v3/nauji2015/Right1_R.off';
> galutinis(input,output);
> makeobj_part(output2,1);
> makeobj_part(output3,2);
> makeoff_part(output4,1);
> makeoff_part(output5,2);
```

Jungtinio C^2 paviršiaus sudarymas iš keturkampių tinklo paviršiaus

```
> restart;

> inicial:=proc(input1)
> local i,j,k,n,m,G;
> global T, S, Sp, Fig;
> G:=readdata(input1,10000);
> Fig:={};
> for i from 1 to G[1][1] do
> T[i]:=G[i+1];
> end do;
> i:=i+1;
> for j from 1 to G[1][2] do
> n:=trunc(G[i][1]);
> for k from 1 to n do
> S[n][j][k]:=trunc(G[i][k+1])+1;
> end do;
> S[n][j]:=convert(S[n][j],list);
> if nops(G[i])=n+4 then
> Sp[n][j]:=[G[i][n+2], G[i][n+3], G[i][n+4]];
> Sp[n][j]:=convert(Sp[n][j],list);
> end if;
> Fig:=`union`(Fig,{n});
> i:=i+1;
> end do;
> for m in Fig do
> Sp[m]:=convert(Sp[m],list);
> S[m]:=convert(S[m],list);
> end do;
> return();
> end proc;

> up:=proc(m,n,k)
> local i;
> i:=`mod`(m+k,n);
> if i=0 then
> return(n);
> else
> return(i);
> end if;
> end proc;

> down:=proc(m,n,k)
> local i;
> i:=`mod`(m-k,n);
> if i=0 then
> return(n);
> else
> return(i);
> end if;
> end proc;

> tinklas:=proc(n)
> local d,m1,m2,p,i,j,k;
> global T1, T2;
> d:=trunc(log[2](n-1))+1;
> m1:=trunc((n+2)/2);
> m2:=n+1-m1;
> for i from 1 to d do
> p:=2^i/n*m2-1;
> for j from 1 to m1 do
> T1[i][m1-j+1]:=1-p-(j-1)*2^i/n;
> end do;
> T1[i]:=convert(T1[i],list);
> for k from 1 to m2 do
> T2[i][m2-k+1]:=1-(k-1)*2^i/n;
> end do;
> T2[i]:=convert(T2[i],list);
> m1:=trunc(m2/2);
> m2:=m2-m1;
```

```
> end do;
> return();
> end proc;

> positions:=proc(n)
> local i,j,k,m;
> global T1, T2, P1, P2;
> m:=0;
> for i from 1 to trunc(log[2](n-1))+1 do
> for j from 1 to nops(T1[i]) do
> P1[i][j]:=j+m;
> end do;
> m:=m+nops(T1[i]);
> for k from 1 to nops(T2[i]) do
> P2[i][k]:=k+m;
> end do;
> P1[i]:=convert(P1[i],list);
> P2[i]:=convert(P2[i],list);
> end do;
> rerutn();
> end proc;

> masks:=proc(gen,n)
> local i,j,k,centras;
> global a;
> for i from 1 to trunc(log[2](n-1))+1 do
> centras:=(1-7/(4*gen))*a[i][1][1][1]+3/(2*gen^2)*sum(a[i][k][1][2],k=1..gen)+
  1/(4*gen^2)*sum(a[i][k][2][2],k=1..gen);
> for j from 1 to gen do
> a[i+1][j][1][1]:=centras;
> a[i+1][j][1][2]:=a[i+1][down(j,gen,1)][2][1];
> a[i+1][j][1][3]:=a[i+1][down(j,gen,1)][3][1];
> a[i+1][j][1][4]:=a[i+1][down(j,gen,1)][4][1];
> a[i+1][j][2][1]:=1/16*(a[i][j][1][2]+a[i][j][2][2]+a[i][up(j,gen,1)][2][1]+
  a[i][up(j,gen,1)][2][2])+3/8*(a[i][j][2][1]+a[i][j][1][1]);
> a[i+1][j][2][2]:=1/4*(a[i][j][1][1]+a[i][j][1][2]+a[i][j][2][1]+a[i][j][2][2]);
> a[i+1][j][2][3]:=1/16*(a[i][j][1][1]+a[i][j][1][3]+a[i][j][2][3]+a[i][j][2][1])+
  3/8*(a[i][j][1][2]+a[i][j][2][2]);
> a[i+1][j][2][4]:=1/4*(a[i][j][1][2]+a[i][j][1][3]+a[i][j][2][2]+a[i][j][2][3]);
> a[i+1][j][3][1]:=1/64*(a[i][j][1][2]+a[i][j][3][2]+a[i][up(j,gen,1)][2][1]+
  a[i][up(j,gen,1)][2][3])+3/32*(a[i][j][1][1]+a[i][j][2][2]+a[i][j][3][1]+
  a[i][up(j,gen,1)][2][2])+9/16*a[i][j][2][1];
> a[i+1][j][3][2]:=1/16*(a[i][j][1][1]+a[i][j][1][2]+a[i][j][3][1]+a[i][j][3][2])+
  3/8*(a[i][j][2][1]+a[i][j][2][2]);
> a[i+1][j][3][3]:=1/64*(a[i][j][1][1]+a[i][j][1][3]+a[i][j][3][1]+a[i][j][3][3])+
  3/32*(a[i][j][1][2]+a[i][j][2][1]+a[i][j][2][3]+a[i][j][3][2])+9/16*a[i][j][2][2];
> a[i+1][j][3][4]:=1/16*(a[i][j][1][2]+a[i][j][3][2]+a[i][j][1][3]+a[i][j][3][3])+
  3/8*(a[i][j][2][2]+a[i][j][2][3]);
> a[i+1][j][4][1]:=1/16*(a[i][j][3][2]+a[i][j][2][2]+a[i][up(j,gen,1)][2][2]+
  a[i][up(j,gen,1)][2][3])+3/8*(a[i][j][2][1]+a[i][j][3][1]);
> a[i+1][j][4][2]:=1/4*(a[i][j][2][1]+a[i][j][3][1]+a[i][j][2][2]+a[i][j][3][2]);
> a[i+1][j][4][3]:=1/16*(a[i][j][2][1]+a[i][j][3][1]+a[i][j][2][3]+a[i][j][3][3])+
  3/8*(a[i][j][2][2]+a[i][j][3][2]);
> a[i+1][j][4][4]:=1/4*(a[i][j][2][2]+a[i][j][2][3]+a[i][j][3][2]+a[i][j][3][3]);
> end do;
> end do;
> return();
> end proc;

> centrinis:=proc(gen,a)
> local i,j,k,b;
> global center;
> for j from 1 to gen do
> b[1][j][1][1]:=a[1][j][1][1];
> b[1][j][1][2]:=a[1][j][1][2];
> b[1][j][2][1]:=a[1][j][2][1];
> b[1][j][2][2]:=a[1][j][2][2];
> end do;
> for i from 1 to 100 do
> center:=(1-7/(4*gen))*b[i][1][1][1]+3/(2*gen^2)*sum(b[i][k][1][2],k=1..gen)+
  1/(4*gen^2)*sum(b[i][k][2][2],k=1..gen);
> for j from 1 to gen do
> b[i+1][j][1][1]:=center;
> b[i+1][j][1][2]:=b[i+1][down(j,gen,1)][2][1];
```

```
> b[i+1][j][2][1]:=1/16*(b[i][j][1][2]+b[i][j][2][2]+b[i][up(j,gen,1)][2][1]+
  b[i][up(j,gen,1)][2][2])+3/8*(b[i][j][2][1]+b[i][j][1][1]);
> b[i+1][j][2][2]:=1/4*(b[i][j][1][1]+b[i][j][1][2]+b[i][j][2][1]+b[i][j][2][2]);
> end do;
> end do;
> return();
> end proc;

> points:=proc(i,j,Q1,Q2,W1,W2)
> local u,v,g,h,m,nn;
> global K,P;
> m:=nops(Q1[i]);
> nn:=nops(Q2[i]);
> for u from 1 to m do
> for v from 1 to nn do
> K[4][j][W1[i][u]][W2[i][v]]:=[0,0,0]:
> for g from 0 to 3 do
> for h from 0 to 3 do
> K[4][j][W1[i][u]][W2[i][v]]:=K[4][j][W1[i][u]][W2[i][v]]+binomial(3, g)*Q1[i][u]^g*
  (1-Q1[i][u])^(3-g)*binomial(3, h)*Q2[i][v]^h*(1-Q2[i][v])^(3-h)*P[g+1][h+1];
> end do;
> end do;
> end do;
> end do;
> return();
> end proc;

> surface:=proc(gen)
> local i,j,h;
> global P,T1,T2,K,num;
> for h from 1 to gen do
> K[4][num[h]][n+1][n+1]:=center;
> end do;
> for i from 1 to trunc(log[2](n-1))+1 do
> for j from 1 to gen do
> weights(a[i+1][j][4][4],a[i+1][j][3][4],a[i+1][j][2][4],a[i+1][j][1][4],a[i+1][j][4][3],
  a[i+1][j][3][3],a[i+1][j][2][3],a[i+1][j][1][3],a[i+1][j][4][2],a[i+1][j][3][2],
  a[i+1][j][2][2],a[i+1][j][1][2],a[i+1][j][4][1],a[i+1][j][3][1],a[i+1][j][2][1],
  a[i+1][j][1][1]);
> points(i,num[j],T1,T1,P1,P1);
> weights(a[i+1][j][3][4],a[i+1][j][2][4],a[i+1][j][1][4],a[i+1][down(j,gen,1)][4][2],
  a[i+1][j][3][3],a[i+1][j][2][3],a[i+1][j][1][3],a[i+1][down(j,gen,1)][3][2],
  a[i+1][j][3][2],a[i+1][j][2][2],a[i+1][j][1][2],a[i+1][down(j,gen,1)][2][2],
  a[i+1][j][3][1],a[i+1][j][2][1],a[i+1][j][1][1],a[i+1][down(j,gen,1)][1][2]);
> points(i,num[j],T1,T2,P1,P2);
> weights(a[i+1][j][4][3],a[i+1][j][3][3],a[i+1][j][2][3],a[i+1][j][1][3],a[i+1][j][4][2],
  a[i+1][j][3][2],a[i+1][j][2][2],a[i+1][j][1][2],a[i+1][j][4][1],a[i+1][j][3][1],
  a[i+1][j][2][1],a[i+1][j][1][1],a[i+1][up(j,gen,1)][2][4],a[i+1][up(j,gen,1)][2][3],
  a[i+1][up(j,gen,1)][2][2],a[i+1][up(j,gen,1)][2][1]);
> points(i,num[j],T2,T1,P2,P1);
> end do;
> end do;
> return();
> end proc;

> briaunos:=proc()
> local i,j,k,u,m,n;
> global T,Fig,S,Br,VidBr,RibBr,VidTsk,RibTsk;
> VidBr:={};
> RibBr:={};
> VidTsk:={};
> RibTsk:={};
> m:=1;
> for i in Fig do
> for j from 1 to nops(S[i]) do
> for k from 1 to i do
> Br[m]:={S[i][j][k],S[i][j][up(k,i,1)]};
> m:=m+1;
> end do;
> end do;
> end do;
> Br:=sort(convert(Br,list));
> for u from 1 to nops(Br)-1 do
> if Br[u]<>Br[u+1] then
```

```
> RibBr:=`union`(RibBr,{Br[u]});
> RibTsk:=`union`(RibTsk,Br[u]);
> else
> VidBr:=`union`(VidBr,{Br[u]});
> u:=u+1;
> end if;
> end do;
> VidTsk:=`minus`({seq(n,n=1..nops(convert(T,list)))},RibTsk);
> return();
> end proc;

> taskukart:=proc()
> local i,br1;
> global Br,Tsk1;
> br1:={Br[]};
> for i from 1 to nops(br1) do
> Tsk1[i]:=br1[i][];
> end do;
> Tsk1:=sort(convert(Tsk1,list));
> return();
> end proc;

> yptaskai:=proc(Tsk1)
> local i,j,k,m;
> global TskBase,Tsk;
> TskBase:={};
> k:=1;m:=nops(Tsk1);
> for i from 1 to m do
> if Tsk1[i]=Tsk1[up(i,m,1)] then
> k:=k+1;
> else
> TskBase:=`union`(TskBase,{k});
> Tsk[k][i]:=Tsk1[i];
> k:=1;
> end if;
> end do;
> for j in TskBase do
> Tsk[j]:=convert(Tsk[j],set);
> if `intersect`(Tsk[j],RibTsk)<>{}
> then printf("\%s\n", "egzistuoja ypatingasis ribinis taskas");
> end if;
> Tsk[j]:=`intersect`(Tsk[j],VidTsk);
> end do;
> return();
> end proc;

> kaimynai:=proc()
> local i,j,p,i1,j1,h,Fig1,index1,index2,q,r,k,m;
> global Fig,S,Gret;
> Fig1:=[seq(S[Fig[p]][],p=1..nops(Fig))];
> m:=0;
> r:=1;
> k:=nops(S[Fig[r]]);
> for q from 1 to nops(Fig1) do
> if q<=k then
> index1[q]:=Fig[r];
> index2[q]:=q-m;
> else
> r:=r+1;
> k:=k+nops(S[Fig[r]]);
> m:=m+nops(S[Fig[r-1]]);
> index1[q]:=Fig[r];
> index2[q]:=q-m;
> end if;
> end do;
> for i from 1 to nops(Fig1) do
> for j from 1 to nops(Fig1[i]) do
> for i1 from i+1 to nops(Fig1) do
> for j1 from 1 to nops(Fig1[i1]) do
> if Fig1[i][j]=Fig1[i1][j1] then
> if Fig1[i][up(j,nops(Fig1[i]),1)]=Fig1[i1][up(j1,nops(Fig1[i1]),1)] then
> Gret[index1[i]][index2[i]][Fig1[i][j]][Fig1[i][up(j,nops(Fig1[i]),1)]]:=
  [index2[i1],seq(Fig1[i1][up(j1,nops(Fig1[i1]),1+h)],h=1..nops(Fig1[i1])-2)];
> Gret[index1[i]][index2[i]][Fig1[i][up(j,nops(Fig1[i]),1)]][Fig1[i][j]]:=
```

```
  [index2[i1],seq(Fig1[i1][down(j1,nops(Fig1[i1]),h)],h=1..nops(Fig1[i1])-2)];
> Gret[index1[i1]][index2[i1]][Fig1[i][j]][Fig1[i][up(j,nops(Fig1[i]),1)]]:=
  [index2[i],seq(Fig1[i][down(j,nops(Fig1[i]),h)],h=1..nops(Fig1[i])-2)];
> Gret[index1[i1]][index2[i1]][Fig1[i][up(j,nops(Fig1[i]),1)]][Fig1[i][j]]:=
  [index2[i],seq(Fig1[i][up(j,nops(Fig1[i]),1+h)],h=1..nops(Fig1[i])-2)];
> else if Fig1[i][up(j,nops(Fig1[i]),1)]=Fig1[i1][down(j1,nops(Fig1[i1]),1)] then
> Gret[index1[i]][index2[i]][Fig1[i][j]][Fig1[i][up(j,nops(Fig1[i]),1)]]:=
  [index2[i1],seq(Fig1[i1][down(j1,nops(Fig1[i1]),1+h)],h=1..nops(Fig1[i1])-2)];
> Gret[index1[i]][index2[i]][Fig1[i][up(j,nops(Fig1[i]),1)]][Fig1[i][j]]:=
  [index2[i1],seq(Fig1[i1][up(j1,nops(Fig1[i1]),h)],h=1..nops(Fig1[i1])-2)];
> Gret[index1[i1]][index2[i1]][Fig1[i][j]][Fig1[i][up(j,nops(Fig1[i]),1)]]:=
  [index2[i],seq(Fig1[i][up(j,nops(Fig1[i]),1+h)],h=1..nops(Fig1[i])-2)];
> Gret[index1[i1]][index2[i1]][Fig1[i][up(j,nops(Fig1[i]),1)]][Fig1[i][j]]:=
  [index2[i],seq(Fig1[i][down(j,nops(Fig1[i]),h)],h=1..nops(Fig1[i])-2)];
> end if;
> end if;
> end if;
> end do;
> end do;
> end do;
> end do;
> return();
> end proc;

> weights:=proc(A11,A12,A13,A14,A21,A22,A23,A24,A31,A32,A33,A34,A41,A42,A43,A44)
> global P;
> P[1][1]:=1/36*(A11+A13+A31+A33)+1/9*(A12+A21+A23+A32)+4/9*A22;
> P[1][2]:=1/18*(A13+A33)+1/9*(A12+A32)+2/9*A23+4/9*A22;
> P[1][3]:=1/18*(A12+A32)+1/9*(A13+A33)+2/9*A22+4/9*A23;
> P[1][4]:=1/36*(A12+A14+A32+A34)+1/9*(A13+A22+A24+A33)+4/9*A23;
> P[2][1]:=1/18*(A31+A33)+1/9*(A21+A23)+2/9*A32+4/9*A22;
> P[2][2]:=1/9*A33+2/9*(A23+A32)+4/9*A22;
> P[2][3]:=1/9*A32+2/9*(A22+A33)+4/9*A23;
> P[2][4]:=1/18*(A32+A34)+1/9*(A22+A24)+2/9*A33+4/9*A23;
> P[3][1]:=1/18*(A21+A23)+1/9*(A31+A33)+2/9*A22+4/9*A32;
> P[3][2]:=1/9*A23+2/9*(A22+A33)+4/9*A32;
> P[3][3]:=1/9*A22+2/9*(A23+A32)+4/9*A33;
> P[3][4]:=1/18*(A22+A24)+1/9*(A32+A34)+2/9*A23+4/9*A33;
> P[4][1]:=1/36*(A21+A23+A41+A43)+1/9*(A22+A31+A33+A42)+4/9*A32;
> P[4][2]:=1/18*(A23+A43)+1/9*(A22+A42)+2/9*A33+4/9*A32;
> P[4][3]:=1/18*(A22+A42)+1/9*(A23+A43)+2/9*A32+4/9*A33;
> P[4][4]:=1/36*(A22+A24+A42+A44)+1/9*(A23+A32+A34+A43)+4/9*A33;
> return();
> end proc;

> nepalankus:=proc()
> local i,j,k,m,p,q,nepal1;
> global Fig,S,Tsk,TskBase,nepal;
> p:=`minus`(TskBase,{4});
> nepal1:={};
> for q in p do
> nepal1:=`union`(nepal1,Tsk[q]);
> end do;
> m:=`minus`(Fig,{4});
> if m<>{} then
> k:=1:
> for i in m do
> for j from 1 to nops(S[i]) do
> nepal[k]:=S[i][j][];
> k:=k+1:
> end do;
> end do;
> nepal:=convert(nepal,set);
> else
> nepal:={};
> end if;
> nepal:=`union`(nepal,nepal1);
> return();
> end proc;

> points1:=proc(i)
> local u,v,g,h;
> global P,K,n;
> for u from 0 to n do
```

```
> for v from 0 to n do
> K[4][i][u+1][v+1]:=[0,0,0]:
> for g from 0 to 3 do
> for h from 0 to 3 do
> K[4][i][u+1][v+1]:=K[4][i][u+1][v+1]+binomial(3, g)*(u/n)^g*(1-u/n)^(3-g)*
  binomial(3, h)*(v/n)^h*(1-v/n)^(3-h)*P[g+1][h+1];
> end do;
> end do;
> end do;
> end do;
> return();
> end proc;

> render1:=proc()
> local i,j,k,t,p1,p2,p3,p4,r1,r2,r3,r4,q1,q2,q3,q4,A11,A12,A13,A14,A21,A22,A23,A24,
  A31,A32,A33,A34,A41,A42,A43,A44,M;
> global T,S,Gret,nepal,P,kasliko,TskBase,TskPr;
> for i from 1 to nops(S[4]) do
> if `intersect`({S[4][i][]},nepal)={} then
> p1:=S[4][i][1]; TskPr[4][i]:=1;
> p2:=S[4][i][2];
> p3:=S[4][i][3];
> p4:=S[4][i][4];
> A22:=T[p1];
> A23:=T[p2];
> A32:=T[p4];
> A33:=T[p3];
> r1:=Gret[4][i][p2][p1][3];q1:=Gret[4][i][p2][p1][1];
> r2:=Gret[4][i][p3][p2][3];q2:=Gret[4][i][p3][p2][1];
> r3:=Gret[4][i][p4][p3][3];q3:=Gret[4][i][p4][p3][1];
> r4:=Gret[4][i][p1][p4][3];q4:=Gret[4][i][p1][p4][1];
> for t from 1 to 8 do
> M[t]:=false;
> end do;
> if nops(Gret[4][i][p1][p2])=3 then
> A12:=T[Gret[4][i][p2][p1][2]];
> A13:=T[r1];
> M[2]:=true;
> if nops(Gret[4][q1][p2][r1])=3 then
> A14:=T[Gret[4][q1][p2][r1][2]];
> M[1]:=true;
> end if;
> else
> A12:=2*A22-A32;
> A13:=2*A23-A33;
> end if;
> if nops(Gret[4][i][p2][p3])=3 then
> A24:=T[Gret[4][i][p3][p2][2]];
> A34:=T[r2];
> M[4]:=true;
> if nops(Gret[4][q2][p3][r2])=3 then
> A44:=T[Gret[4][q2][p3][r2][2]];
> M[3]:=true;
> end if;
> else
> A24:=2*A23-A22;
> A34:=2*A33-A32;
> end if;
> if nops(Gret[4][i][p3][p4])=3 then
> A43:=T[Gret[4][i][p4][p3][2]];
> A42:=T[r3];
> M[6]:=true;
> if nops(Gret[4][q3][p4][r3])=3 then
> A41:=T[Gret[4][q3][p4][r3][2]];
> M[5]:=true;
> end if;
> else
> A43:=2*A33-A23;
> A42:=2*A32-A22;
> end if;
> if nops(Gret[4][i][p4][p1])=3 then
> A31:=T[Gret[4][i][p1][p4][2]];
> A21:=T[r4];
> M[8]:=true;
```

```
> if nops(Gret[4][q4][p1][r4])=3 then
> A11:=T[Gret[4][q4][p1][r4][2]];
> M[7]:=true;
> end if;
> else
> A31:=2*A32-A33;
> A21:=2*A22-A23;
> end if;
> if M[1]=false then
> if M[2]=true and M[8]=true then
> A11:=A21-A31/2+A12-A13/2;
> else if M[2]=true and M[8]=false then
> A11:=2*A12-A13;
> else A11:=2*A21-A31;
> end if;
> end if;
> end if;
> if M[3]=false then
> if M[4]=true and M[2]=true then
> A14:=A13-A12/2+A24-A34/2;
> else if M[4]=true and M[2]=false then
> A14:=2*A24-A34;
> else A14:=2*A13-A12;
> end if;
> end if;
> end if;
> if M[5]=false then
> if M[6]=true and M[4]=true then
> A44:=A34-A24/2+A43-A42/2;
> else if M[6]=true and M[4]=false then
> A44:=2*A43-A42;
> else A44:=2*A34-A24;
> end if;
> end if;
> end if;
> if M[7]=false then
> if M[8]=true and M[6]=true then
> A41:=A42-A43/2+A31-A21/2;
> else if M[8]=true and M[6]=false then
> A41:=2*A31-A21;
> else A41:=2*A42-A43;
> end if;
> end if;
> end if;
> weights(A11,A12,A13,A14,A21,A22,A23,A24,A31,A32,A33,A34,A41,A42,A43,A44);
> points1(i);
> else
> kasliko[i]:=i;
> end if;
> end do;
> kasliko:=convert(kasliko,set);
> if nops(TskBase)<>1 then
> featurender();
> end if;
> return();
> end proc;

> preprint:=proc()
> local f,g,i,j,k,buvo,u,v,h,did;
> global S,ss,Gret,TskPr,atimti,GG,eil,n;
> did:=0;
> for k from 1 to nops(S[4]) do
> buvo[k]:=false;
> end do;
> atimti[0]:=1;
> for i from 1 to nops(S[4]) do
> buvo[i]:=true;
> f:=TskPr[4][i];
> for j from 1 to 4 do
> g[j]:=Gret[4][i][S[4][i][up(f,4,j-1)]][S[4][i][up(f,4,j)]];
> if nops(g[j])=1 or buvo[g[j][1]]=false
> then ss[i][j]:=0 else ss[i][j]:=1;
> end if;
> end do;
```

```
> atimti[i]:=atimti[i-1]+(n+1)^2-(n-ss[i][1]-ss[i][3]+1)*(n-ss[i][2]-ss[i][4]+1);
> for u from 1+ss[i][1] to n+1-ss[i][3] do
> for v from 1+ss[i][4] to n+1-ss[i][2] do
> GG[i][u][v]:=did;
> did:=did+1;
> end do;
> end do;
> if ss[i][1]=1 then
> posukis(g[1][1],g[1][2]);
> for h from 1 to n+1 do
> GG[i][1][h]:=eil[h];
> end do;
> end if;
> if ss[i][2]=1 then
> posukis(g[2][1],g[2][2]);
> for h from 1 to n+1 do
> GG[i][h][n+1]:=eil[h];
> end do;
> end if;
> if ss[i][3]=1 then
> posukis(g[3][1],g[3][2]);
> for h from 1 to n+1 do
> GG[i][n+1][n-h+2]:=eil[h];
> end do;
> end if;
> if ss[i][4]=1 then
> posukis(g[4][1],g[4][2]);
> for h from 1 to n+1 do
> GG[i][n-h+2][1]:=eil[h];
> end do;
> end if;
> end do;
> return();
> end proc;

> posukis:=proc(i,pr)
> local t,dd;
> global n,eil,atimti,GG;
> dd:=1;
> while S[4][i][dd]<>pr do
> dd:=dd+1;
> end do;
> if dd=2 then
> for t from 1 to n+1 do
> eil[t]:=GG[i][n+1][t];
> end do;
> else if dd=1 then
> for t from 1 to n+1 do
> eil[t]:=GG[i][n-t+2][n+1];
> end do;
> else if dd=4 then
> for t from 1 to n+1 do
> eil[t]:=GG[i][1][n-t+2];
> end do;
> else
> for t from 1 to n+1 do
> eil[t]:=GG[i][t][1];
> end do;
> end if;
> end if;
> end if;
> return();
> end proc;

> makeoff:=proc()
> local i,j,k,a,b,c;
> global K,mesh,Sp,ss,Gret,TskPr,atimti,GG;
> preprint();
> fopen(mesh,WRITE,TEXT):
> fprintf(mesh,"%s\n%s\n%d %d %d\n","# created by akatasis","OFF",nops(S[4])*(n+1)^2-
  atimti[nops(S[4])]+1,nops(S[4])*n^2,nops(S[4])*(n+1)^2)-atimti[nops(S[4])];
> for i from 1 to nops(S[4]) do
> for j from 1+ss[i][1] to n+1-ss[i][3] do
> for k from 1+ss[i][4] to n+1-ss[i][2] do
```

```
> fprintf(mesh,"%f %f %f\n",K[4][i][j][k][1],K[4][i][j][k][2],K[4][i][j][k][3]);
> end do;
> end do;
> end do;
> for k from 1 to nops(S[4]) do
> a:=Sp[4][k][1]:b:=Sp[4][k][2]:c:=Sp[4][k][3]:
> for i from 1 to n do
> for j from 1 to n do
> fprintf(mesh,"%d %d %d %d %f %f %f\n",4,GG[k][i][j],GG[k][i][j+1],GG[k][i+1][j+1],
  GG[k][i+1][j],a,b,c);
> end do;
> end do;
> end do;
> fclose(mesh):
> return();
> end proc;

> featurender:=proc()
> local i,j,k,t,q,yp;
> global S,Tsk,TskBase,kasliko,kasliko2,re;
> for q in kasliko do
> re[q]:=false;
> end do;
> kasliko2:={};
> yp:={};
> for t in `minus`(TskBase,{4}) do
> yp:=`union`(yp,Tsk[t]);
> end do;
> for i in kasliko do
> k:=`intersect`({S[4][i][]},yp);
> if nops(k)>=2 then
> kasliko2:=`union`(kasliko2,k);
> else if nops(k)=1 and re[i]=false then
> feature1(i,k[]);
> end if;
> end if;
> end do;
> return();
> end proc;

> feature1:=proc(p,q)
> local m1,m2,m3,f,i,k,h,t,korekt,tink;
> global S,T,Gret,a,gen,TskBase,Tsk,re,num,TskPr;
> k:=p;
> for t in `minus`(TskBase,{4}) do
> if q in Tsk[t] then
> gen:=t;
> end if;
> end do;
> korekt:=false;tink:=true;
> f:=1;
> while q<>S[4][k][f] do
> f:=f+1;
> end do;
> i:=1;
> while korekt=false do
> re[k]:=true;num[i]:=k;
> a[1][i][1][1]:=T[q];m1:=S[4][k][up(f,4,1)];m2:=S[4][k][up(f,4,2)];
  m3:=S[4][k][up(f,4,3)];TskPr[4][k]:=down(f,4,2);
> a[1][i][2][1]:=T[m1];
> a[1][i][2][2]:=T[m2];
> a[1][i][1][2]:=a[1][down(i,gen,1)][2][1];
> a[1][i][1][3]:=a[1][down(i,gen,1)][3][1];
> h:=Gret[4][k][m1][m2];
> if nops(h)=3 then
> a[1][i][3][2]:=T[h[2]];
> a[1][i][3][1]:=T[h[3]];
> else if nops(h)=1 then
> a[1][i][3][2]:=2*a[1][i][2][2]-a[1][i][1][2];
> a[1][i][3][1]:=2*a[1][i][2][1]-a[1][i][1][1];
> else korekt:=true;tink:=false;
> end if;
> end if;
> h:=Gret[4][k][m2][m3];
```

```
> if nops(h)=3 then
> a[1][i][2][3]:=T[h[3]];
> else if nops(h)=1 then
> a[1][i][2][3]:=2*a[1][i][2][2]-a[1][i][2][1];
> else korekt:=true;tink:=false;
> end if;
> end if;
> h:=Gret[4][h[1]][m2][h[3]];
> if nops(h)=3 then
> a[1][i][3][3]:=T[h[2]];
> else if nops(h)=1 then
> a[1][i][3][3]:=2*a[1][i][2][2]-a[1][i][1][1];
> else korekt:=true;tink:=false;
> end if;
> end if;
> k:=Gret[4][k][q][m1][1];
> f:=1;
> while q<>S[4][k][f] do
> f:=f+1;
> end do;
> if i=gen then korekt:=true;
> end if;
> i:=i+1;
> end do;
> if tink=true then
> centrinis(gen,a);
> masks(gen,n):
> tinklas(n):
> positions(n):
> surface(gen);
> a:='a':
> gen:='gen':
> end if;
> return();
> end proc;

> clear:=proc()
> global T,S,Sp,Fig,T1,T2,P1,P2,a,center,K,P,num,VidBr,RibBr,VidTsk,RibTsk,Br,
  Tsk1,Tsk,Gret,TskBase,nepal,n,mesh,kasliko,kasliko2,gen,re;
> T:='T'; S:='S'; Sp:='Sp'; Fig:='Fig';
> T1:='T1'; T2:='T2'; P1:='P1'; P2:='P2';
> a:='a'; center:='center'; K:='K'; P:='P';
> num:='num'; VidBr:='VidBr'; RibBr:='RibBr';
> VidTsk:='VidTsk'; RibTsk:='RibTsk'; Br:='Br';
> Tsk1:='Tsk1'; Tsk:='Tsk'; Gret:='Gret';
> TskBase:='TskBase'; nepal:='nepal';
> n:='n'; mesh:='mesh'; kasliko:='kasliko';
> kasliko2:='kasliko2'; gen:='gen'; re:='re';
> end proc;

> visi:=proc()
> inicial(input1);
> briaunos();
> taskukart();
> yptaskai(Tsk1);
> kaimynai();
> nepalankus();
> render1();
> preprint();
> makeoff();
> clear();
> end proc;

> input1:=`C:/Users/akatasis/Desktop/nulis.off`:
> mesh:=`C:/Users/akatasis/Desktop/patch.off`:
> n:=10;
> visi();
```

Martynas Sabaliauskas

NESTANDARTINĖS AVALYNĖS GAMYBOS FORMŲ PAVIRŠIŲ KOMPIUTERINIO MODELIAVIMO TECHNOLOGIJA

Daktaro disertacija

Technologijos mokslai, informatikos inžinerija (07 T)

Redaktorė Audra Ivanauskienė