**Vilniaus Universitetas**

**Matematikos ir informatikos fakultetas, Kompiuterijos katedra**

**Joris Mykolas Medeisis**

**Multirezoliucinio dvinarinio erdvės dalinimo medžio sudarymo algoritmas**

**(Multiresolution Binary Space Partition Tree Algorithm, angl.)**

**2017.12**

**Įžanga**

Šioje programos dokumentacijoje bus aptarimas „Multiresolution modeling using binary space partitioning trees” straipsnyje[[1]](#footnote-1) išdėstyta (Multirezolution Binary Space Partition tree) duomenų struktūros sandara bei šios sruktūros sudarymas. Remiantis šiuo straipsniu buvo pabandyta programiškai įgyvendinti šios duomenų struktūros sukonstravimo algoritmą.

Pačios dokumentacijos tiklas – programinio įgyvendinimo procesas bei algoritmo analizė lyginant pavykusią algoritmo implementaciją su straipsnio turiniu, aptraiant iškilusius sunkumus bei tam tikrus neatitikimus.

**Algoritmo taikymas**

Šis algoritmas dvimatei erdvei buvo pristatytas 1997 metais „Binary Space Partittioning Trees: A Multiresolution Approch“ Katalonijos Jaume I universiteto publikuotame straipsnyje[[2]](#footnote-2). Dėl tuometinių technikos galimybiu silpnumo, buvo reikalingi prekalkuliaciniai algoritmai, kurie paruoštu objektu atvaizdavima skirtingomis rezoliucijomis, kurios būtų naudojamas žiūrovui nuo objekto esant skirtingu nuotolio. Kitaip sakant – būtų optimizuotas objekto perspektyvoje atvaizdavimas.

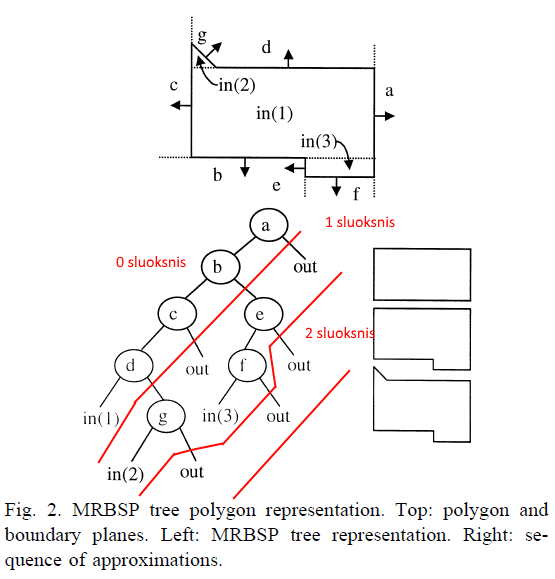
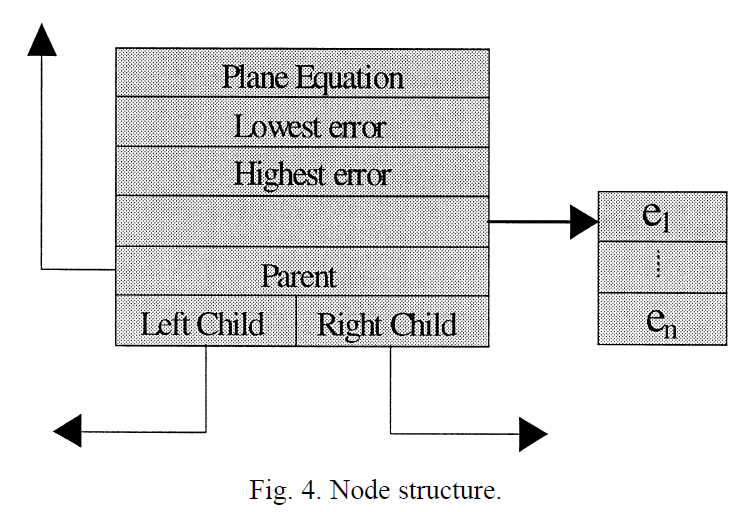
Pagrindinė idėja: kiekienam objektui (poligonui) paruosti jo atvaizdavimo medį (duomenų struktūrą), kurį pasitelkiant, būtų nusprendžiama kaip detaliai reiktų atvaizduoti poligoną, jog kompiuterio technikos galimybės būtų išnaudotos optimizuotai, vaizdo kokybė nenukentėtų. Pavyzdžiui, namo objektas turi kelis šimtus laužčių, kraštinių, taškų, žiūrint į šį objektą, tarkim virtualioje erdvėje iš kelių šimtų metrų perspektyvos, tos detalės tampa nebeatvaizduojamos, dėl pixelių tankio apribojimo, kitaip sakant objektas turi daugiau grafinės informacijos (taškų), negu jam skiriama pixelių, todėl ta informacija persikloja ir grafinis procesorius (GPU), turi atlikti naudos neduodanti veiksma – nupiesti duotame pixelyje, kelis skirtingus taškus, kurie tiesiog užkloja vienas kitą.

Ši problematika pasiulyta spresti, pasitelkiant išanksto pasiruoštą schema (medį), kuriame būtų nurodyta, kokiam esant atstumui, kurias kraštines piešti, kad vaizdas neprarastų kokybės, o GPU nebūtų apkraunamas nereikalingu darbu.

**MRBSP medžio sandara**

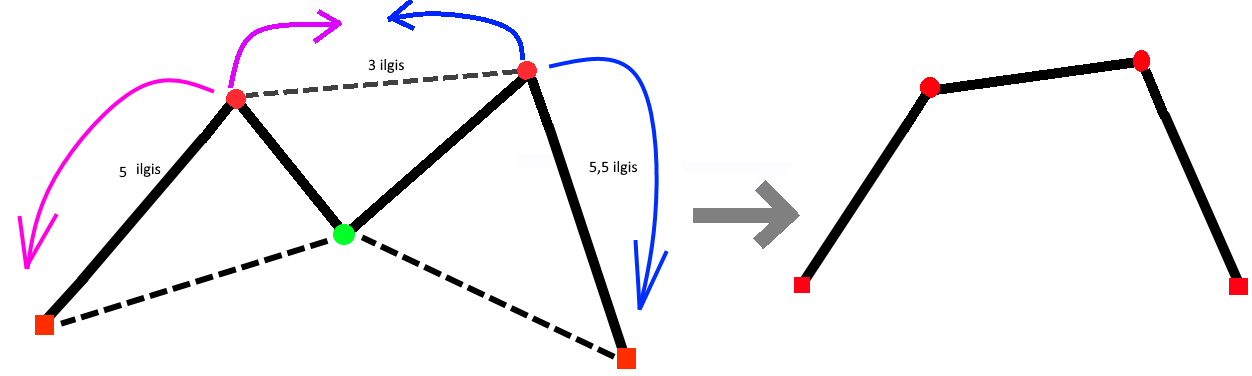
Medis susideda iš mazdgų, iš kurių kiekviename yra talpinama nuoroda į kraštinių sąrašą, kurios pirklauso vienai tiesiai – aproksimacijos bendrai kraštinei. Taip pat yra talpinama papildoma informacija kaip identifikacijos numeris medyje, apatinė ir viršutinė riba, kurios nusako kokiame nuotolio intervale šis medžio mazgas bus piešiamas bei nuorodos į tėvinį bei du vaikinius (In ir Out) mazgus. „In“ vaikas – žymi, jog vaikinė aproksimacijos kraštinė yra aproksimacijos viduje, Out – aproksimacijos išorėje. Taip pat reiktų paminėti, jog tikroji reikšmė (nuotolio intervalas) pagal, kuria piešiamos krastinės priklausančios vienam mazgui yra talpinamos ne mazge, o kiekvienoje is kraštinių, kurios priklauso mazgui.

Pats MRBSP medis yra išbalansuotas į kairę pusę, todėl jo analizavimui ir piešimui patogiau žymėti, ne jo aukštį, o sluoksnius, kurie įvedami nuo kairiausios medžio atšakos, kurioje talpinami, labiausiai orginalaus objekto viduje esancios aproksimacijų kraštinės.

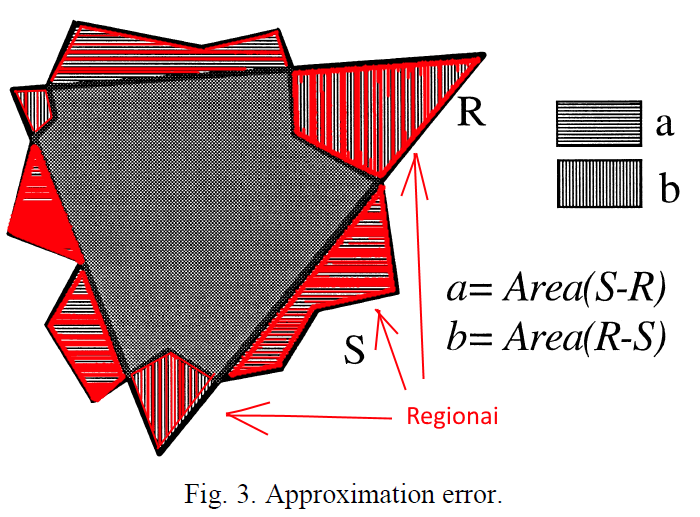


**Medžio sudarymo algoritmas**

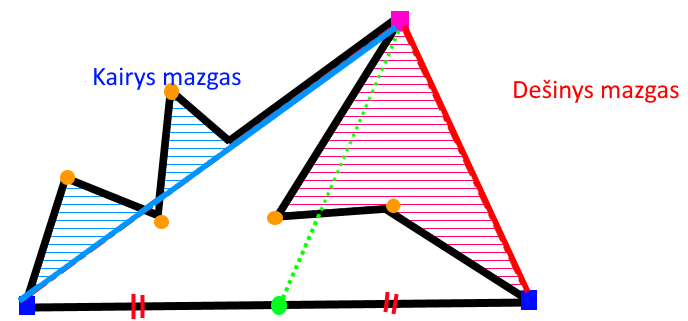
1. Sudaroma pradinė abstrakcija, kuri randama iteruojant orginalu poligona per aproksimavimo algoritmą, kuris susideda:
2. Penkių orginalaus poligono viršūnių: kairiausios, kairesnės, vidurinės, dešinesnės, dešiniausios.
3. Apskaičiuojami atstumai tarp kairesnės ir dešinesnės – viduriomenamoji aproksimavimo kraštinė(AK), tarp kairiausios ir kairesnės – kairioji kraštinė, tarp dešinesnės ir dešiniausios –dešinioji kraštinė, viršunių
4. Lyginami gauti rezultatai, pirma, jei vidurio AK yra trumpiausia iš visų trijų rastų kraštinių ilgių – panaikinamas vidurio taškas, ir tarp kairesnės ir desinesnės viršunių nubrėžiama tiesė. Jei ne, tada lyginamos kairioji ir dešinioji kraštinės, jei kairioji trumpesnė, naikinamas kairesnioji viršūnė bei kairiausioji ir vidurinioji viršunės sujungiamos kraštine. Jei trumpiausia iš visų dešinioji kraštinė, tada naikinamas dešinesnė viršunė ir dešiniausioji sujungiama kraštine su viduriniaja viršune.
5. Aproksimavimo algoritmas tęsiamas, kol lieka 3 kraštinės.



1. Gautai pirminei aproksimacija – sukuriami pirmieji trys mazgai, kurie visi priklauso nuliniam sluoksniui, ju matomumo paklaida [0.0 – 1.0] ty. Nuo 0% iki 100%.
2. Randamas pradinis regionų sąrašas pasitelkiant geometrinę aproksimacijos ir orginalau poligonų sankirtą. Kiekvienas regiono objektas žino apie savo viršūnes, bazinę kraštinę (pagrindą), medžio mazgą, kas atitinka aproksimacijos ištisą kraštinę (gali sutapti su bazinės kraštinės ribomis).



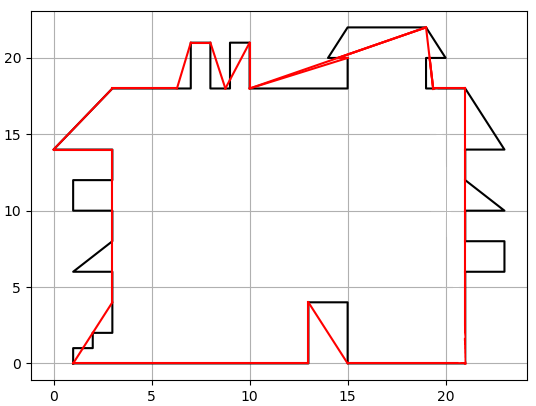
1. Įėjimas į while ciklą:
2. Regionų sąrašas surušiuojamas pagal plotą, nuo didziausio iki mažiausio.
3. Didžiausias regionas išimamas iš sąrašo ir iš jo pagaminama regiono aproksimacija - laužtė, kurią sudaro try viršunės ir dvi kraštinės.
4. Ant bazinės kraštinės randamas vidurio taškas.
5. Paskui randama regiono viršūnė toliausiai nutolusi nuo vidurio taško neiskaitant viršunių, kurios žymi bazinės kraštinės galus.
6. Nubrėžiamos kratinės tarp toliausiai nutolusios viršunės ir abiejų bazinių viršunių – gaunama regiono aproksimacija.
7. Atliekama geometrinės sankirtos operacija tarp regiono ir gautos regiono aproksimacijos

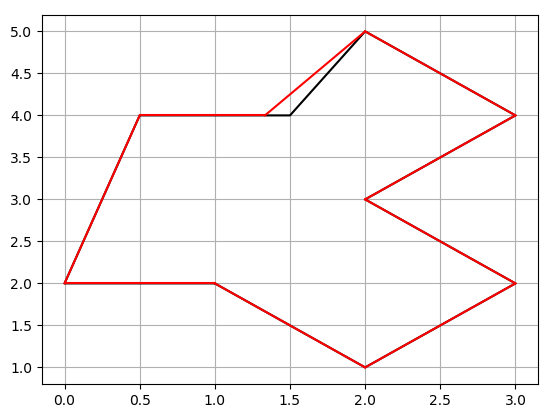


1. Per sankirtos taškus žymimos naujų – mažų regiono regionėlių bazinės pradžios ir pabaigos.
2. Į pagrindinę programą gražinama regiono aproksimacija, bei jos kairiosios kraštinės ir dešiniosios kraštinės atskyri regionų sąrašai.
3. Iš regiono aproksimacijos abiejų krašinių gaunamį du nauji mazgai, kurie įdedami į medį pagal tai kokiam (tėviniam) mazgui, priklausė regiono bazinė kraštinė. Jei regionas buvo vidurje einamosios poligono aproksimacijos, tada viena kraštinė tampa „In“ vaiku, o antroji kraštinė tampa pirmosios vaiku „In“, jei regionas buvo išorėje einamosios poligono aproksimacijos viena kraštinė tampa tėvo „Out“ vaiku (keliauja į sekantį sluoksnį), o antroji kraštinė tampa katik įdėtos kraštinės „In“ vaiku.
4. Kairiesiams regionėliams regionėliams suteikiama informacija apie katik į medį įdėtus tevinius mazgus ir tėvinę kraštinę, tuo tarpu naujai idėtiems mazgams, atnaujinamas kraštinių sąrašas, mazgas turi tiek kraštinių, kiek jame yra regionų.
5. Tiek kairieji, regioneliai, tiek dešinieji regioneliai sudedami į globalų poligonų regionų sąrašą ir kitoje iteracijoje bus lygiaverčiais regionai lyginant kartu su kitais.
6. Regiono aproksimacija prijungiama prie esamos poligono aproksimacijos
7. Apskaičiuojama aproksimacijos paklaida pagal formulę SUM(visų einamųjų regionų plotai) / Orginalaus poligono ploto.
8. Visoms regiono aproksimacijos paveiktoms kraštinėms ir jų mazgui suteikiama žemoji matomumo riba kaip einamoji aproksimacijos paklaida.
9. Visų naujai gautų kraštinių ir jų mazgų (kairiojo ir dešiniojo) aukščiausia matomumo riba nustatoma kaip einamoji aproksimacijos paklaida.
10. Tikrinama ar dar liko regionų regionų sąraše, jei liko – ciklas tęsiamas, jei ne – ciklas užbaigiamas ir gražinamas turimas medis (objektas su dvimačiu masyvu, kurio pirmoji dimensija nurodo medžio sluoksnį, antroji – mazgo eilės numerį sluoksnyje).
11. Gražinamas medis

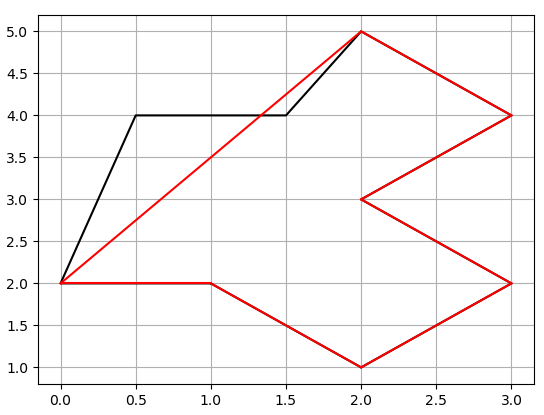
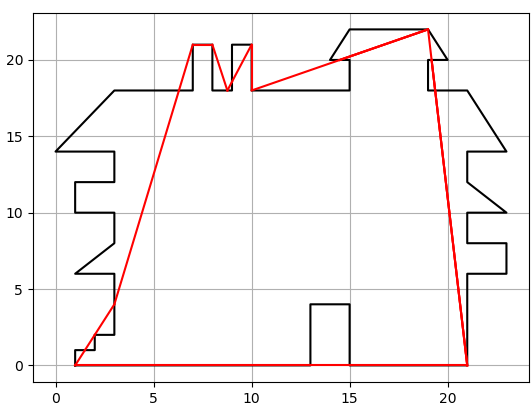
**Rezultatai:**

Namas ir nesudėtingas poligonas, kai objektas nutoles 10% savo matomumo ribos (matomumo paklaidos slenkstis = 0.1):

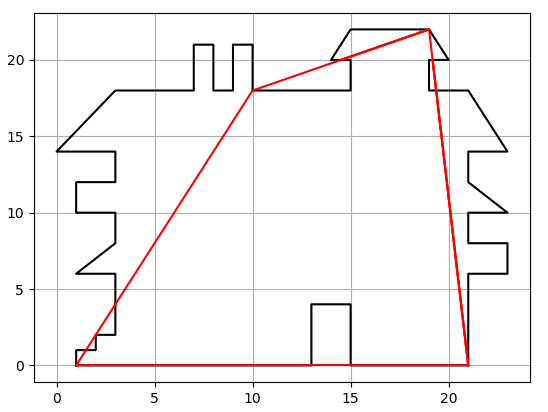
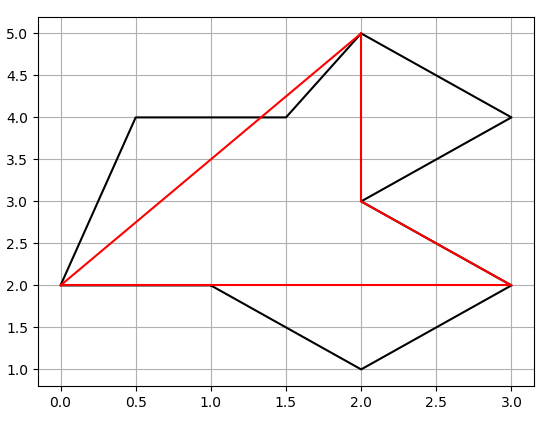




Namas ir nesudėtingas poligonas, kai objektas nutoles 30% savo matomumo ribos (matomumo paklaidos slenkstis = 0.3):

****

Namas ir nesudėtingas poligonas, kai objektas nutoles 80% savo matomumo ribos (matomumo paklaidos slenkstis = 0.8):



**Išvados**

* MRBSP medžio sukonstravimo algoritmas buvo programiškai įgyvendintas pasitelkiant Python 3.6 programavimo kalbą su išroine mathplotlab grafiniui atvaizdavimui skirta biblioteka. Programos valdymo skydelį galima rasti programos pabaigoje po komentaru „### Atlikimas ###”.
* Bandant implementuoti „Multiresolution modeling using binary space partitioning trees” straipsnyje[[3]](#footnote-3) aprašytą MRBSP medžio konstrukcijos algoritmą iškilo tam tikrų neaiškumų, dėl kurių siek tiek teko keisti algoritmo įgyvendinimą, tačiau pavyko pasiekti tenkinamą dvimačio medžio funkcionalumą.
* Pirmiausia, straipsnyje nebuvo iki galo aiškiai išaiškintas pradinės poligono aproksimacijos radimo algoritmas. Apart paieškos tarp kairesniosios ir dešinesniosios trumpiausių kaimyninių kraštinių lyginant su įsivaizduojama kraštine paieškos, kai panaikinama vidurinioji viršūnė (žiūrėti 1.3 algoritmo punktą), kitų atvėjų sprendiniai nebuvo pateikti, todėl teko improvizuoti įvedant dar du atvejus, kai panaikinama kairesnioji ar dešinenioji viršūnė ir įsivaizduojama kraštinė atsiranda ne per vidurį, o kairėje arba dešinėje. Taip pat nebuvo iki galo paaiškinta kokiomis sąlygomis galutinė aproksimacija gali turėti daugiau nei tris viršūnes, todėl buvo pasilikta išskirtinai prie tribriaunės galutinės aproksimacijos visais atvejais.
* Straipsnyje taip pat nebuo iki galo išaiškintas regiono aproksimacijos radimo algoritmas, todėl laisvai interpretuojant buvo įvestas vidurio taškas bazinėje kraštinėje bei matuojami atstumai tarp šio taško ir regiono viršunių neįskaitant bazinei kraštinei priklausančiu dviejų viršunių. Taip pat teko įvesti naujas sąvokas kaip kairieji ir dešinieji tevinio regiono aproksimacijos renionėliai.
* Kadangi nebuvo iki galo paaiškinta kaip reiktų teisingai įterpti naują mazgą į jau egzistuojantį medį, todėl buvo sukurta nauja sąvoka – medžio sluoksnis, kuris indikavo mazgų atstovaujamų kraštinių erdvinį pasiskirstymą einamosios aproksimacijos atzvilgiu (išorinis sluoksnis – mazesniu numeriu, vidinis sluoksnis didesniu – slenkant link medžio dešinės).
* Taip pat buvo priimtas sprendimas kitaip apskaičiuoti einamosios matymo aproksimacijos slenkstį pagal naujai išvestą formulę (SUM(visų einamųjų regionų plotai) / Orginalaus poligono ploto), kurios rezultatai iš dalies sutampa su straipsnyje pateiktais rezultatais. Šis sprendimas buvo priimtas, kadangi straipsnyje iki galo nebuvo pagrįstas loginis ryšys tarp sąokų Approximation Error, Initial Error, Current Error, Intersection Area a, Intersection Area b, Polygon Area apskaičiuojant matomumo paklaidą, kurios esmę sudarė plotų santykis. Todėl buvo nuspręsta remtis pagrindiniu kriterijumi – kiek santykinai aproksimacija neatitinka orginalaus poligono ploto ir formos.

**Šaltiniai:**

Pagrindinis:

* „Multiresolution modeling using binary space partitioning trees” Joaqu´ın Huerta, Miguel Chover, Jose´ Ribelles, Ricardo Quiro´s. Computer Networks and ISDN Systems 30. *Computer Science Department, Jaume I Uni*Í*ersity, Castello*´*n, Spain.* 1998*.*

Papildomi:

* „Binary Space Partittioning Trees: A Multiresolution Approch“ J. Huerta, M. Chover, R Quiros, R. Vivo, J Ribelles; *Computer Science Department, Jaume I University. Castellon, Spain.* 1997.

1. „Multiresolution modeling using binary space partitioning trees” Joaqu´ın Huerta, Miguel Chover, Jose´ Ribelles, Ricardo Quiro´s. Computer Networks and ISDN Systems 30. *Computer Science Department, Jaume I Uni*Í*ersity, Castello*´*n, Spain. 1998.* [↑](#footnote-ref-1)
2. „Binary Space Partittioning Trees: A Multiresolution Approch“ J. Huerta, M. Chover, R Quiros, R. Vivo, J Ribelles; Computer Science Department, Jaume I University. Castellon, Spain. 1997. [↑](#footnote-ref-2)
3. „Multiresolution modeling using binary space partitioning trees” Joaqu´ın Huerta, Miguel Chover, Jose´ Ribelles, Ricardo Quiro´s. Computer Networks and ISDN Systems 30. [↑](#footnote-ref-3)