Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Boštjan Čotar

Adaptivna segmentacija 3D volumnov

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Saša Divjak
Somentor: doc. dr. Matija Marolt

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi preučite postopke za segmentacijo 3D volumnov. Implementirajte metodo, ki temelji na rekurzivnem deljenju vokslov in tako bolje zajame podrobnosti modela. Integrirajte jo v orodje za 3D vizualizacijo NeckVeins. Prav tako metodo združite z obstoječo metodo Marching Cubes in izvedite primerjavo glede na zgradbo in natančnost rekonstruiranih 3D modelov.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Boštjan Čotar, z vpisno številko **63090038**, sem avtor diplomskega dela z naslovom:

Adaptivna segmentacija 3D volumnov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Saše Divjaka in somentorstvom doc. dr. Matije Marolta,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 16. septembra 2014 Podpis avtorja:

Za začetek bi se rad zahvalil vsem, ki so pomagali pri nastanku dela. Najprej se zahvaljujem somentorju doc. dr. Matiji Maroltu za pomoč pri nastanku dela. Velika zahvala gre as. mag. Cirilu Bohaku, ki mi je predstavil tematiko naloge in tekom njenega nastanka usmerjal, svetoval in pomagal.

Iskreno se zahvaljujem vsem prijateljem, ki so popestrili moje življenje v času študija. Marko, Andrej, Uroš, Črt, Lea in Matej, hvala za igričarske in družabne večere. Hvala moji punci, Moniki, ki me je podpirala in verjela vame, hvala tudi za slike.

Nenazadnje bi se rad zahvalil svoji družini. Mama in tata, hvala za vso podporo v času študija, predvsem v 1. letniku. Hvala bratu za vse obilne večerje, ki si jih pripravil, ne bi si mogel želeti boljšega cimra.

Svojim staršem.

Kazalo

Povzetek

Cilj diplomske naloge je razviti metodo za segmentacijo 3D modela drevesne strukture žil za program NeckVeins. S primerjavo meje praga in vrednosti vsakega posameznega oglišča voksla, iz 3D matrike določimo, kateri voksli predstavljajo telo drevesne strukture žil, kateri predstavljajo ozadje in kateri predstavljajo tako imenovane »sosede«. Za bolj natančen izris 3D modela, je v metodi implementirano rekurzivno deljenje sosednjih vokslov na osem manjših segmentov. Za vsakega od manjših segmentov se ponovno določi, v katero kategorijo spada. Na ta način dobimo model drevesa žile, zgrajenega iz vokslov različnih velikosti. V diplomskem delu je tako na novo razviti algoritem implementiran v delovanje programa NeckVeins, ki že vsebuje svojo metodo za segmentacijo. Na koncu je prikazana primerjava vseh algoritmov za segmentacijo v programu NeckVeins. Za primerjavo razlik med algoritmi se uporabljajo zaslonske slike 3D modelov ožilja. Primerjamo zgradbo žil, stopničavost in natančnost izrisa.

Ključne besede: segmentacija, 3D model, Marching Cubes, voksel.

Abstract

The main objective of the thesis was to develop a method for segmentation of a 3D vascular tree model for the program NeckVeins. By comparing the threshold value with the value of each vertex of the segment, which is a voxel, the method determines which voxels represent the main body of the vein tree, which represent the background and which represent the so-called »neighbours«. Depending on how detailed we want to plot the 3D model, the method recursively divides the adjacent voxels to eight smaller voxels. For each of the smaller segment we re-determine to which category it belongs, so that in the end the vein tree model is built from voxels of different sizes. The newly developed algorithm is implemented in the program NeckVeins, which itself already contains its own segmentation method. We compare the differences between the algorithms using screenshots of the 3D vascular model, paying attention to the structure of the blood vessel, staircasing and the accuracy of the plot.

Keywords: segmentation, 3D model, Marching Cubes, voxel.

Poglavje 1

Uvod

Živimo v času, ko je računalniška tehnologija ključnega pomena. S pomočjo računalnikov si lahko olajšamo marsikatero opravilo. Zato ni čudno, da se računalniška tehnologija pojavlja v vseh panogah. Ena izmed takih panog je tudi zdravstvo. Ko omenimo zdravstvo, mislimo predvsem na moderno medicino. Že od začetka razvoja, ki sega v konec 19. stoletja, sta moderna medicina in tehnologija neločljivi. Tako je z odkritjem rentgenskih žarkov in razvojem naprav za slikanje zgradbe telesa prišel največji napredek na področju vizualizacije. Za vpogled v telo ni bilo potrebno več odpirati telesa. Računalniška tehnologija je šla še korak dlje in danes imamo ne le 2D ampak celo 3D slike organov, kosti ali ožilja. Delo s tako sofisticirano tehnologijo zdravnikom bistveno olajša postavitev ustrezne diagnozo in načina zdravljenja. Prav zato je bila možnost dela na projektu, ki bo v bodoče mogoče pomagal in poenostavil delo zdravnikom, ter morda tudi komu rešil življenje, bila glavna motivacija pri izbiri področja in teme za diplomsko nalogo.

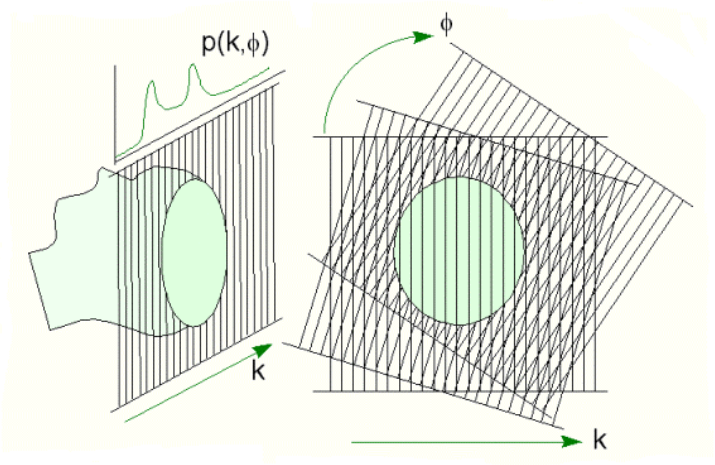
Danes poznamo veliko zvrsti medicinskih slik, ki jih uporabljamo za različne namene. Tako poznamo ultrazvočne slike, rentgenske slike, slike z jedrsko magnetno resonanco, slike nuklearne medicine in svetlobne slike. Največja razlika med zvrstmi slik je način zajemanja in prikaza podatkov. Ultrazvočne slike nastajajo s pomočjo ultrazvoka, kar ni nič drugega kot odboj zvoka, kjer

se spremeni gostota snovi. Računalniki s pomočjo pizioelektričnih kristalov, ki pretvarjajo zvočno energijo v električno in obratno, rekonstruirajo sliko. Pridobljene slike so slabe kvalitete, je pa tak način zajemanja slik povsem neškodljiv za človeško telo. Pri rentgenskih slikah so osnova rentgenski žarki, ki nastanejo v posebnih ceveh. Imamo več načinov zajemanja slik s pomočjo rentgenskih žarkov. Poznamo klasičen način (Slika 1.1), kjer dobimo sliko na flourescentnem filmu, na kateremu je slika z izmenjujočim odtenkom sive barve, ki gre od črne do bele barve. Na ta način lahko za ustrezen odtenek določimo ali je plin, maščoba, tekočina, kost ali kovina.

Slika 1.1: Primer rentgenske slike dlani. Povzeto po [6]

Če želimo slike z večjim kontrastom, uporabimo digitalno subtrakcijsko angiografijo (angl. Digital Substraction Angiography, v nadaljevanju DSA). Pacientu se vbrizga kontrast, ki poveča absorpcijo v krvnih žilah. S pomočjo računalnika nato odštejemo dobljeno sliko od osnovne. Če želimo pridobiti 3D model žil, pa uporabimo računalniško tomografijo (angl. Computed Tomography, v nadaljevanju CT). Tako kot za ostali dve metodi, so tudi pri tej metodi osnova rentgenski žarki. Z rotacijo sistema okoli objekta (Slika 1.2) dobimo več slik. Tako iz zaporednih (2D slik) računalnik izračuna prerez (3D

sliko) s tako imenovano rekonstrukcijo iz projekcij (angl. back-projection reconstruction). Vse tri metode za pridobitev slik slonijo na uporabi tehnologije rentgenskih žarkov, ki so za človeka nevarni.



Slika 1.2: Rekonstrukcija iz projekcij. Povzeto po [13]

Zaradi vse večje zaskrbljenosti in vprašanja, kako izpostavljanje rentgenskim žarkom vpliva na človekovo zdravje, so kmalu po začetku uporabe CT odkrili nov način slikanja z uporabo jedrske magnetne resonance. Slikanje z magnetno resonanco (angl. Magnetic Resonance Imaging, v nadaljevanju MRI) za človeka ni škodljivo in rezultati so slike visoke kakovosti. Nekatera atomska jedra se obnašajo kot magneti z lastnim magnetskim poljem. Pulzi pravokotnega elektromagnetnega valovanja povzročijo njihovo precesijo in indukcijo toka v merilni tuljavi. Tok v merilni tuljavi je sorazmeren gostoti magnetov, tako lahko različna tkiva ločimo po tej gostoti. Podobno kot pri CT, se tudi v tem primeru uporablja tehnika rekonstrukcije iz projekcij.

Pri slikanju z nuklearno medicino se uporabljajo radioaktivni izotopi, ki oddajajo gama žarke. Če smo pri prejšnjih načinih slikanja s pomočjo zunanjih naprav generirali rentgenske žarke za sevanje v telo, gre pri nuklearni medicini za sevanje iz telesa. Gama žarke opazujemo s pomočjo gama kamere.

Z obračanjem kamere, različnimi izotopi in tehnikami slikanja dobimo 3D slike. Najbolj znani tehniki slikanja sta SPECT (angl. Single photon emission CT) in PET (angl. Positron emission tomography).

Zadnji način slikanja je svetlobno slikanje. Svetlobno slikanje se opravlja z endoskopom, to je tehnični inštrument za pregled notranjosti telesnih votlin in organov. Temu posegu pravimo endoskopija in poznanih je več vrst. Uporablja se zlasti pri preiskavah prebavne cevi, žil in notranjih organov.

Zajem in vizualizacija 3D slik je pomembna tudi za druga področja, kot na primer za geologijo. S pomočjo MRI tehnologije lahko določamo prepustnost kamnin za ogljikovodike. Rentgenske žarke se uporablja tudi v fiziki, za kristalografijo (ugotavljanje razporeditev atomov, molekul in ionov v trdnih snoveh). Rezultate metod tako procesirajo računalniki, ki zgradijo ustrezne modele in slike. Celotno področje vizualizacije z uporabo računalnikov je dokaj mlado področje, ki se močno opira na napredek v računalništvu. V tako mladem področju je še veliko prostora za spremembe in izboljšave.

V diplomski nalogi smo se ukvarjali z razvojem metode za segmentacijo, ki bo iz vhodnih podatkov o 3D posnetku ustvarila ustrezen 3D model. Koncept metode smo vzeli iz metode, opisane v [10] in ga za namen dela z voksli prilagodili. Pri delu sem spoznal nove tehnologije, kot sta OpenGL in LWJGL, in algoritme za segmentacijo, kot sta Marching Cubes in Multi-level Partition of Unity Implicits.

Diplomsko delo je sestavljeno iz treh delov. V prvem delu je predstavljena tehnologija o vizualizaciji tridimenzionalnih podatkov, predvsem na medicinskem področju in algoritmi, ki poskrbijo za njeno učinkovito delovanje. V drugem delu sledi opis orodij, metod in delovanje programa NeckVeins. V zadnjem delu je predstavljen opis metode, faze razvoja in implementacija. Za lažje ugotavljanje uspešnosti in primerjavo razvitega algoritma z ostalimi algoritmi za segmentacijo v programu, so podane zaslonske slike.

Poglavje 2

Pregled področja

Računalniška tehnologija je ključnega pomena, ko govorimo o vizualizaciji tridimenzionalnih (3D) podatkov, natančneje vaskularnih struktur. Vizualizacija struktur ožilja, ki predstavljajo eno izmed najbolj kompleksnih in zapletenih struktur človeškega telesa, spada danes med najpomembnejše pridobitve moderne medicine. S pomočjo medicinskih aplikacij, ki uporabljajo to tehnologijo, lahko učinkoviteje prikazujemo 3D slike vaskularnih struktur, živčnih poti ali celo organov. S takim načinom dela zdravniki lažje odkrivajo in postavljajo diagnoze žilnih obolenj, ki jih uvrščamo v sam vrh vzrokov smrti [17, 16]. Takšni statistični podatki še dodatno motivirajo razvijalce k razvoju boljših in bolj natančnih tehnologij za 3D vizualizacijo vaskularnih drevesnih struktur.

Za uspešno postavljanje diagnoz je uporabljena tehnologija ključnega pomena. Ta poskrbi za natančen, visoko kakovosten izris modela ožilja. Podatke za izris ožilja pridobimo s pomočjo naprav, kot so naprava za slikanje z magnetno resonanco (MRI) ali naprava za slikanje z računalniško tomografijo (CT). Te slike računalniško obdelamo in rezultat so volumski posnetki struktur. Nad temi posnetki lahko nato izvajamo metode in algoritme za ekstrakcijo in za segmentacijo.

Z metodami za ekstrakcijo lahko iz volumskih posnetkov pridobimo določene strukture, kot so organi in drevesne strukture žil. Obstaja veliko takih me-

tod, zato je na to temo napisanih veliko člankov. Eden od takih člankov je [7], kjer so predstavljene tehnike za ekstrakcijo primerjane med seboj. Kot ciljno množico tehnik za ekstrakcijo je avtor vzel krvne žile in nevrovaskularne strukture. Poleg tehnik za ekstrakcijo je govora tudi o metodi za segmentacijo cevastih predmetov, ki imajo podobne značilnosti kot žile. V namen preučevanja metod in tehnik za segmentacijo, so te razdeljene v 6 kategorij. Za primerjavo kategorij med seboj so uporabljeni različni kriteriji.

S podobno tematiko se srečamo v [4], kjer avtor poudari, kako pomembni sta postali tehnologiji zajemanja slik s pomočjo rentgenskih žarkov in angiografije. Kljub temu, da je uporaba angiografije postala najbolj zanesljiva metoda za odkrivanje žilnih obolenj, je pri ekstrakciji žil prisotnih še veliko problemov. Glavni tematiki članka sta tako raziskava tehnik za ekstrakcijo in raziskava za izboljšavo ekstrakcije. Predstavljeni so tudi najbolj pomembni algoritmi za segmentacijo žil.

S pregledom metod za segmentacijo se srečamo tudi v [9]. V članku je rahel poudarek na tehnikah segmentacije, ki se jih izvaja nad slikami, zajetimi s 3D načinom slikanja s kontrastom (MRI in CT). Uporabljen je zanimiv pristop primerjanja tehnik med seboj. Analiza poteka vzdolž treh osi, kjer vsaka od osi (model, funkcija, sistem za ekstrakcijo) igra ključno vlogo za izvedbo učinkovite in natančne metode za segmentacijo žil.

Ko govorimo o vizualizaciji volumnov, moramo poudariti, da gre za eno najbolj zanimivih in hitro rastočih področij znanstvene vizualizacije. Do danes je bilo razvitih že veliko različnih tehnik za vizualizacijo. Kot pojasnjeno v [5], večina teh tehnik izhaja iz enega od približno petih temeljnih algoritmov.

Dve taki večji skupini metod za vizualizacijo sta neposredno upodabljanje volumna in posredno upodabljanje volumna. Pri neposrednih metodah, opisanih v [8], podatki za vizualizacijo niso pretvorjeni v nobeno drugo obliko. Posredne metode upodabljanja nadalje delimo na upodabljanje volumna brez modela in upodabljanje volumna z modelom. Za obe metodi je značilno to, da podatke pretvorijo v drugačno obliko. Primerjava neposredne in posredne

metode upodabljanja je opisana v [12].

Pri obravnavi slik v medicini je ocena kakovosti slike glavna skrb. Za razliko od števila člankov, ki se ukvarjajo z metodami za vizualizacijo volumnov, je število člankov, ki se ukvarjajo s kakovostjo končne slike bistveno manj. V [11] je pregledana validacija vizualizacij volumnov v zdravstvu. Namen članka je razviti metodo klasifikacije, ki nam bo pomagala odgovoriti na vprašanje kako in katere parametre izbrati, da bomo izboljšali slikanje in postopke vizualizacije volumnov.

V programu NeckVeins je implementiranih več algoritmov za segmentacijo, kot sta na primer algoritem Marching Cubes in Gaussova metoda za glajenje. Marching Cubes, implementiran po [3], je eden najbolj znanih algoritmov na področju računalniške grafike. Njegova uporaba je v zdravstvu daleč najbolj razširjena za vizualizacijo podatkov zajetih z magnetno resonanco ali računalniško tomografijo. Drugi algoritem je Gaussova metoda za glajenje v treh dimenzijah [19]. Gaussovo glajenje, ali drugače rečeno Gaussova meglica, je rezultat meglenja slike z Gaussovo funkcijo. Je pogosta metoda, ki se uporablja v grafičnih programih. Njen namen je zmanjšanje šuma in podrobnosti slike.

V sami diplomski nalogi se v večini časa ukvarjamo z metodami za segmentacijo v povezavi z metodo za določanje minimalnega praga. S pomočjo segmentacije v volumskem posnetku za posamezne manjše dele - voksle, preverimo njihovo vrednost, glede na nastavljen prag in določimo, ali jih bomo uporabili za prikaz 3D modela ali ne. Potrebno je poudariti, da naloga algoritmov za segmentacijo ni boljši prikaz modela, ampak izbira pravih segmentov za prikaz. Ena od morebitnih situacij, pri katerih lahko pride do nastanka problemov, so tanke strukture žil. Takemu problemu se izognemo z nadaljnjo delitvijo na še manjše segmente, s čimer bolje ocenimo njihovo obliko [10].

Danes lahko 3D modele že opazujemo s pomočjo naprav v modernih bolnicah in klinikah. Prestop iz 2D slike modela v 3D sliko bistveno izboljša delo zdravnikov in olajša prepoznavanje anomalij, kot na primer stenoz in

anevrizmov. Celotno področje vizualizacije z uporabo računalnikov je revolucionaliziralo medicino in lahko pričakujemo še več napredkov.

Poglavje 3

Orodja in metode

3.1 Java in Eclipse

Za razvoj svoje metode za segmentacijo je bil uporabljen programski jezik Java skupaj z razvojnim okoljem Eclipse.

3.1.1 Java

Java je objektno usmerjen, močno tipiziran programski jezik, ki je bil razvit v podjetju Sun Microsystems [20]. Razvit je bil kot zamenjava za programski jezik C++, zato sta si sintaksi obeh jezikov podobni. Java spada med visokonivojske programske jezike, zato pri izvajanju programov prevaja (angl. compile) v nižjenivosjki programski jezik. Njena posebnost je, da se programska koda javinih programov ne prevede v strojno kodo, ampak se ta prevede v bajtno kodo (angl. bytecode). Vse skupaj teče na javanskem virtualnem stroju (angl. Java Virtual Machine - JVM), kar omogoča programerjem izvajanje kode na poljubni platformi, brez potrebe po ponovnem pisanju programa.

3.1.2 Eclipse

Eclipse [18] je večjezično programsko razvojno okolje napisano v javi. S pomočjo vtičnikov lahko v Eclipsu razvijamo tudi v drugih programskih jezikih. Tako kot se za integrirano razvojno okolje (IDE) spodobi, ima Eclipse veliko orodij, kot so razhroščevalnik, prevajalnik, interpreter, urejevalnik kode, ki dopolnjuje izraze in opozarja na napake, ter še mnogo drugih. Pomembna posebnost Eclipsa je ta, da deluje na različnih platformah, bodisi je to Windows, Linux ali MacOS.

3.2 OpenGL

OpenGL (angl. Open Graphics Library) [2] je standardni računalniški programski vmesnik (API) za prikazovanje 2D in 3D grafičnih podob. Običajno se OpenGL uporablja za interakcijo z grafično procesno enoto (GPE), s čimer dosežemo pospešeno strojno upodabljanje. Pred nastankom standarda OpenGL je vsako podjetje, ki se je ukvarjalo z razvojem grafičnih aplikacij, moralo na novo napisati grafični del aplikacije za vsako platformo. Z OpenGL lahko aplikacija ustvari enake efekte na vseh platformah, ki uporabljajo katero koli grafično kartico s podporo za OpenGL.

Za delo s standardom OpenGL se uporabljajo knjižnice, ena od najbolj znanih in uporabljena v namen diplomske naloge je knjižnica LWJGL (Lightweight Java Game Library) [1]. LWJGL je odprtokodna javanska programska knjižnica. Visoko zmogljivost knjižnice se običajno uporablja pri razvoju računalniških iger in multimedijskih predstavitev. Poleg OpenGL, LWJGL omogoča razvijalcem dostop do zmogljivih knjižnic kot so OpenCL (angl. Open Computing Language) in OpenAL (angl. Open Audio Library).

3.3 Program NeckVeins

NeckVeins je program, napisan v Javi, namenjen prikazovanju 3D modelov. Primarno je bil razvit za namene vizualizacije 3D modelov žil, vendar se lahko uporablja tudi za vizualizacijo drugih 3D modelov. Za komunikacijo z grafično kartico uporablja programski vmesnik OpenGL. Ima preprost grafični vmesnik (Slika 3.1), ki je zgrajen s pomočjo knjižnice TWL. Pogled na 3D model je možno spreminjati s spreminjanjem zornega kota in položaja modela.

Slika 3.1: Slika preprostega grafičnega vmesnika.

Aplikacija NeckVeins je nastala v okviru diplomske naloge Vizualizacija žil tilnika z OpenGL, katere avtor je Simon Žagar [15]. V njegovi diplomski nalogi je implementiral grafični vmesnik, možnost obračanja modela in premikanja kamere. Poleg tega je implementiral še osvetljevanje in senčenje z uporabo senčilnikov. Aplikacija NeckVeins je bila uporabljena tudi v diplomski nalogi Segmentacija prostorskih medicinskih podatkov na GPE, avtorja Anžeta Sodje

[14]. Njegov doprinos k že obstoječi aplikaciji je bila implementacija segmentacijskih algoritmov. Za hitrejše izvajanje segmentacije je s pomočjo knjižnice OpenCL implementiral izvajanje na grafično procesni enoti.

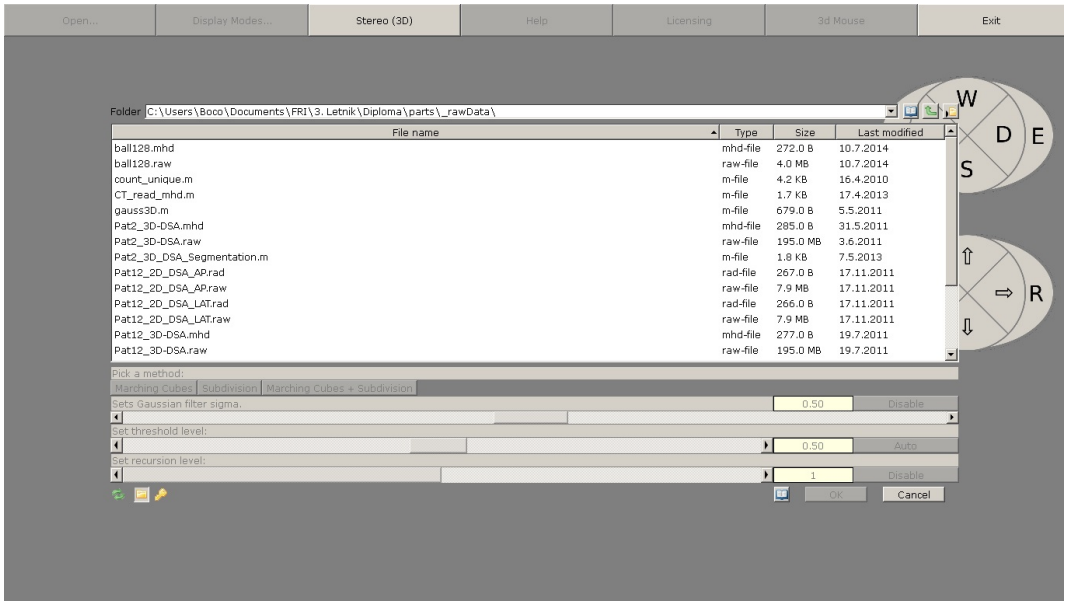
3.3.1 Delovanje

Samo delovanje programa Neck Veins je razdeljeno v tri faze: branje vhodnih podatkov, segmentacija in izris modela. Vhodni podatki so lahko iz različnih formatov. Za našo nalogo smo uporabljali format `.mhd`, iz katerega smo zgradili 3D model, sestavljen iz vokslov, kjer ima vsako vozlišče svojo vrednost. Po izbiri datoteke z vhodnimi podatki in obdelavi podatkov v 3D model, sledi segmentacija. V okviru segmentacije se izberejo določene vrednosti parametrov, kot so minimalna meja praga ali vrednost sigma parametra Gaussovega filtra, ki pomaga pri odpravljanju šuma. Po končani segmentaciji se v programu izriše 3D model drevesne strukture žil, ali druge oblike, če smo za vhodne podatke vzeli 3D model nečesa drugega.

Za lažjo interakcijo s programov je uporabnikom na voljo uporabniški vmesnik. Za njegovo gradnjo je bila uporabljena namenska knjižnica TWL (angl. Themable Widget Library). Uporabniku se ob zagonu programa odpre aplikacija z menijem in navigacijskimi paneli (Slika 3.1).

S klikom na gumb `Open...` se v programu odpre novo okno, kjer izbiramo vhodno datoteko in parametre segmentacije (Slika 3.2).

Po segmentaciji se nam na zaslonu izriše 3D model iz vhodnih podatkov. Pogled na model lahko upravljamo z navigacijskimi paneli (Slika 3.3) v obliki elipse, ki se nahajajo na desni strani aplikacije, ali s tipkovnico. Tako lahko model približamo, oddaljimo ali obračamo.



Slika 3.2: Okno za izbiranje vhodne datoteke.

Slika 3.3: Z rumeno barvo sta obkrožena panela za upravljanje kamere.

Poglavje 4

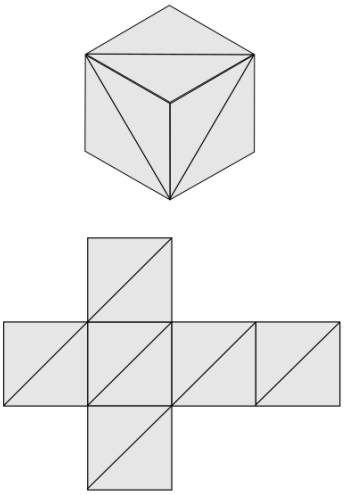
Adaptivna segmentacija 3D volumnov

4.1 Opis metode

Glavna naloga diplomskega dela je bila implementacija novega algoritma za segmentacijo. Zaradi lažje implementacije v obstoječi program NeckVeins je algoritem napisan v programskem jeziku Java. Program ima implementirane že nekatere algoritme za segmentacijo in metodo za izrisovanje. Osnovni gradnik, s katerim program gradi 3D model, je voksel, ki ima 8 oglišč. Vsaka ploskev voksla je sestavljena iz dveh trikotnikov. Program jemlje iz izhodnih podatkov našega algoritma, to je iz seznama vozlišč (angl. ArrayList), po tri oglišča in izrisuje trikotnike. Vsaka površina kocke oziroma voksla je tako zgrajena iz 12 trikotnikov (Slika 4.1).

4.1.1 Vhod in izhod

Preden se lotimo opisa delovanja algoritma je potrebno definirati, kaj točno je vhod in kaj izhod algoritma. Pri tem velja poudariti dejstvo, da se tako vhod kot izhod algoritma razlikujeta od vhoda in izhoda programa NeckVeins.



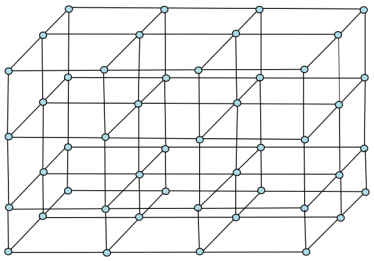
Slika 4.1: Zgradba enega voksla in njegova mreža.

Vhodni podatki za program NeckVeins so datoteke vrste `.obj` ali `.mhd`. Datoteka formata `.obj` vsebuje podatke o vsakem vozlišču (angl. vertex), normalah, teksturah in ploskvah, ki jih opredeljujemo kot mnogokotnik, sestavljen iz seznama vozlišč. Datoteka formata `.mhd` vsebuje meta podatke (tip objekta, število dimenzij matrike, velikost dimenzij, pozicijo, orientacijo ipd.) o 3D posnetku. Izhod programa NeckVeins pa je izrisan 3D model.

Vhodni podatki našega algoritma pa niso datoteke, temveč parametri, kot so meja praga in globina rekurzije, in 3D matrika vokslov. Slednja je zgenerirana iz vhodne datoteke programa NeckVeins še pred začetkov izvajanja algoritma za segmentacijo. Matrika (Slika 4.2) je zgrajena iz vokslov in vsak voksel ima 8 oglišč. Vsako oglišče ima določeno vrednost in na podlagi vrednosti vseh oglišč, ki gradijo voksel, bomo le tega uvrstili v ustrezno skupino vokslov. Oglišča tistih vokslov, uvrščenih v skupino vokslov, ki predstavlja telo modela, so dodana v seznam. Izhod algoritma, ob koncu izvajanja se-

gmentacije, je seznam oglišč. Ta seznam bo uporabil program NeckVeins za izrisovanje trikotnikov 3D modela.

Po preučitvi vhodov in izhodov bi lahko rekli, da je vhod našega algoritma odvisen od vhodne datoteke programa NeckVeins in izhod programa NeckVeins je odvisen od izhoda našega algoritma.



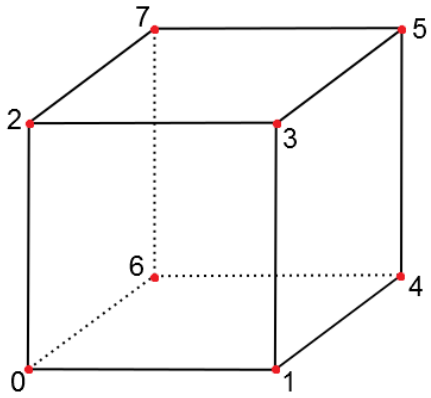
Slika 4.2: Zgradba na novo zgenerirane 3D matrike vokslov.

4.1.2 Delovanje metode

Naš algoritem ima glavno metodo, znotraj katere se za določene voksle izvede rekurzivna metoda deljenja na manjše segmente. Sprva je potrebno omeniti, da ima glavna metoda določene parametre, to so 3D matrika vokslov, minimalna meja praga in maksimalna globina rekurzije. Na začetku glavne metode smo inicializirali določene spremenljivke, kot je začetna globina rekurzije, sezname in slovarje. Poleg teh parametrov je bilo potrebno definirati zgradbo voksla (Slika 4.3). Kot smo že omenili, je vsak voksel zgrajen iz 12 trikotnikov, zato smo v tabeli definirali trojice oglišč, ki sestavljajo teh 12 trikotnikov (odsek kode 4.1).

Listing 4.1: Tabela oglišč

```
int indeksi[] = {
    0, 2, 1, 2, 3, 1,
    1, 3, 4, 3, 5, 4,
    4, 5, 6, 5, 7, 6,
    6, 7, 0, 7, 2, 0,
    6, 0, 4, 0, 1, 4,
    2, 7, 3, 7, 5, 3
};
```



Slika 4.3: Zgradba na novo zgenerirane 3D matrike vokslov.

Po inicializaciji vseh parametrov je sledilo sprehajanje po ustvarjeni 3D matriki vokslov. Sproti smo za vsak voksel osnovne velikosti $1 \times 1 \times 1$ pregledali vrednosti vseh 8 oglišč. Vrednost vsakega oglišča smo primerjali z minimalno mejo praga. Za vsako oglišče, ki je imelo manjšo vrednost v primerjavi z minimalno mejo praga, smo povečali vrednost spremenljivke *pogoj* za 1. Po končanem primerjanju vseh 8 oglišč, smo v odvisnosti od vrednosti spremenljivke *pogoj* naredili eno od treh možnih akcij:

***pogoj* = 8:** voksel predstavlja ozadje. Tak voksel ignoriramo, se vrnemo na začetek in vzamemo v obdelavo nov voksel;

***pogoj* = 0:** voksel predstavlja telo modela. Njegova oglišča v trojicah vstavimo v seznam oglišč (vrstice $17 - 22$ v psevdokodi 1);

$1 \leq$ *pogoj* < 8: voksel predstavlja tako imenovane »sosede«. Tak voksel gre v obdelavo rekurzivni metodi za deljenje na manjše segmente, katere psevdokoda je na koncu podpoglavja (psevdokoda 1).

Rekurzivna metoda preneha z izvajanjem, ko je njena globina večja ali enaka maksimalni globini rekurzije. Po končani rekurzivni metodi se ponovno vrnemo na začetek in izberemo nov voksel za obdelavo. Postopek se ponavlja dokler ne predelamo celotne 3D matrike vokslov. Na koncu nam kot rezultat glavna metoda vrne seznam oglišč, s katerim bo lahko program NeckVeins izrisoval trikotnike.

```
   Data: tabela z oglišči
 1 globina = globina + 1;
 2 for številka oglišča a manjša od števila oglišč v kocki do
 3     for številka oglišča b manjša od števila oglišč v kocki do
 4         if številka oglišča a razlikuje od števila oglišča b then
 5             izračunaj koordinate središčne točke med sosednjima
               ogliščema a in b z uporabo formule za mediano;
 6             izračunaj vrednost novega oglišča z uporabo povprečne
               vrednosti oglišč a in b;
 7             vstavi novo oglišče v slovar;
 8         else
 9             vstavi podatke o obstoječem oglišču a v slovar;
10         end
11     end
12     for vsa oglišča v kocki do
13         if vrednost oglišča manjša od meje praga then
14             pogoj = pogoj + 1;
15         end
16     end
17     if vrednost spremenljivke pogoj enaka 0 then
18         for vseh 12 trikotnikov kocke do
19             definiraj prvo, drugo in tretje oglišče;
20             dodaj oglišča v vrstnem redu v seznam oglišč;
21         end
22     end
23     if vrednost spremenljivke pogoj različna od 0 then
           // kocko delimo na manjše segmente s klicom rekurzije
24         klic rekurzivne metode;
25     end
26 end
27 globina = globina − 1;
```

Algorithm 1: Psevdokoda rekurzivne metode

4.2 Razvoj metode

Po načrtovanju delovanja algoritma smo se lotili programiranja. Za lažje razumevanje delovanja in boljše planiranje smo razvoj algoritma razdelili na več faz:

- Faza 1 predstavlja sprehod preko oglišč v matriki iz katerih sproti ustvarimo kocke. Prav tako določimo katere kocke spadajo v skupino kock, ki bodo izrisane in kocke, ki spadajo v tako imenovano skupino »sosedov«;
- Faza 2 predstavlja delitev kock, ki so opredeljene kot »sosedi«, na manjše kocke, če je potrebno tudi večkrat;
- Faza 3 predstavlja združitev v enotno metodo z rekurzijo, kjer z vrednostjo indeksa rekurzije določimo njeno globino.

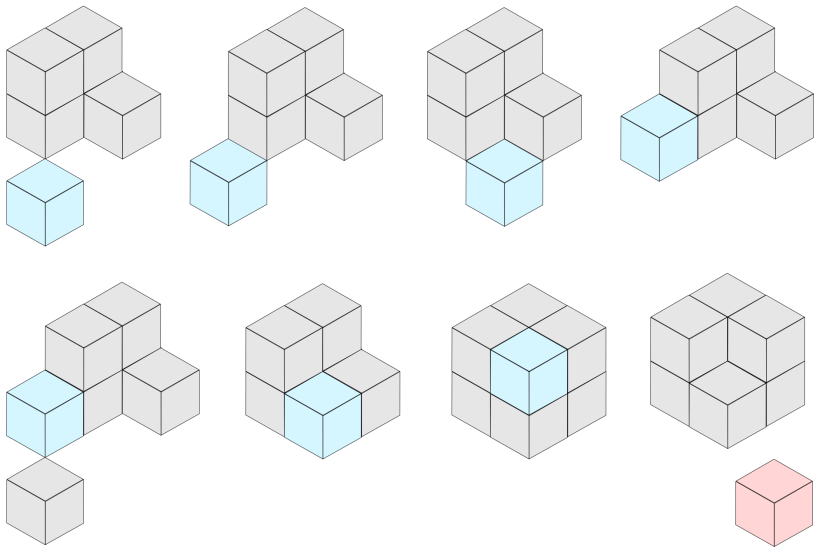
Pri samem razvoju in testiranju delovanja algoritma smo kot vhodne podatke uporabljali 3D model krogle velikosti $128 \times 128 \times 128$, predvsem zaradi hitrejšega izračunavanja in izrisovanja.

Faza 1

V začetni fazi je bilo potrebno se sprehoditi čez celotno 3D matriko oglišč in sproti ustvariti kocke velikosti $1 \times 1 \times 1$. Za vsako na novo ustvarjeno kocko je bilo potrebno primerjati vrednost vsakega posameznega oglišča kocke z vrednostjo meje za minimalno dovoljeni prag. Po primerjanju vseh 8 oglišč so sledili trije možni scenariji:

- vseh 8 oglišč je imelo manjšo vrednost, kot je bila vrednost meje praga (Slika 4.4 - kocka rdeče barve);
- vseh 8 oglišč je imel enako ali večjo vrednost, kot je bila vrednost meje praga (Slika 4.4 - kocka sive barve);

- najmanj 1 in največ 7 oglišč je imelo manjšo vrednost, kot je bila vrednost meje praga (Slika 4.4 - kocka modre barve).



Slika 4.4: Vse možne postavitve sosednjih kock (modra barva) in kock, ki predstavljajo ozadje (rdeča barva), glede na telo žile (siva barva).

V prvem primeru, kjer so bile vrednosti vseh osmih oglišč manjše od meje praga, je kocka predstavljala ozadje modela. V drugem primeru, kjer so bile vse vrednosti oglišč enake ali večje od meje praga, je sledilo dodajanje ploskev v seznam oglišč. Vsako ploskev kocke je bilo tako potrebno razdeliti na 2 trikotnika, skupno torej 12 trikotnikov. Oglišča teh 12 trikotnikov se je nato dodalo v seznam oglišč, iz katerega se je kasneje bralo podatke za izris. V zadnjem primeru so bile take kocke obravnavane kot »sosede« kockam, iz drugega primera.

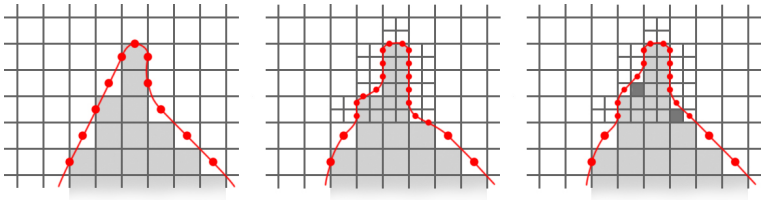
Faza 2

Za razliko od kock, ki so predstavljale ozadje in kock, ki so predstavljale skelet ogrodja, so sosednje kocke nekje vmes. Podobno kot je opisano v [10], smo

sosednje kocke začeli razdeljevati na manjše kocke. Tako smo iz ene kocke velikosti $1 \times 1 \times 1$ pridobili osem kock velikosti $0.5 \times 0.5 \times 0.5$. Pri samem procesu razdeljevanja smo za novo ustvarjene kocke ustvarili nova oglišča z novimi vrednostmi. Za ta namen smo uporabili formulo za iskanje razpolovišča v 3D prostoru med dvema točkama (4.1) in tako pridobili nove koordinate oglišč. Za vrednost novo nastalega oglišča pa smo vzeli aritmetično sredino vrednosti oglišč, med katerima je nastalo novo oglišče.

$$M = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}, \frac{z_1 + z_2}{2} \right) \tag{4.1}$$

Za vsako novo nastalo kocko manjših dimenzij smo primerjali še vrednosti oglišč z vrednostjo meje praga. Tako je vsaka kocka zopet prešla postopek opisan v Fazi 1. Za vse »sosednje« kocke, je tako spet sledilo deljenje na manjše in manjše kocke (Slika 4.5).

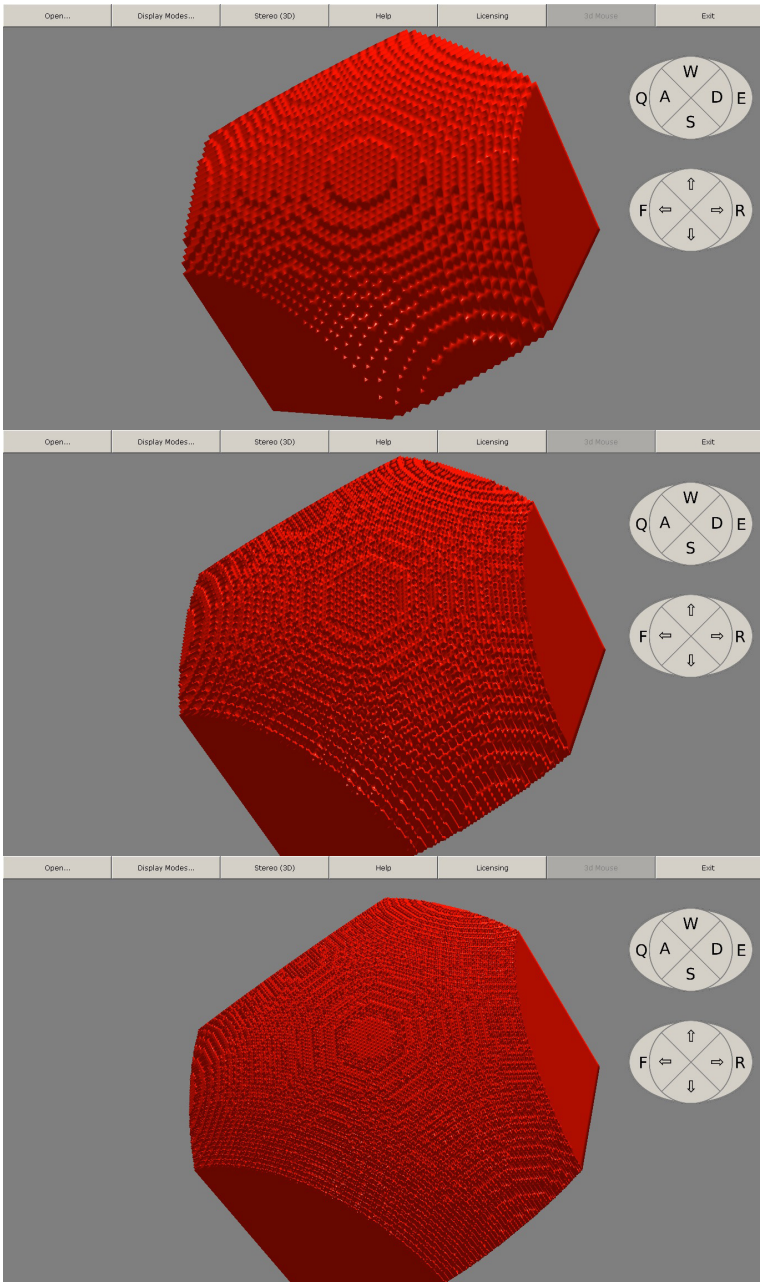


Slika 4.5: Primer delitve kvadratov na manjše kvadrate, za boljši izris obrobe in preprečitev stopničavosti. Povzeto po [10].

Faza 3

Kot je razvidno iz prejšnje faze, se sosednje kocke delijo v manjše. S tem procesom razbijanja kock gradimo podrobnejši model. Preprečujemo predvsem nastanek stopničastih obrob na modelu. Tak način delovanja algoritma je idealen za uporabo rekurzije. Z določitvijo globine rekurzije algoritmu povemo,

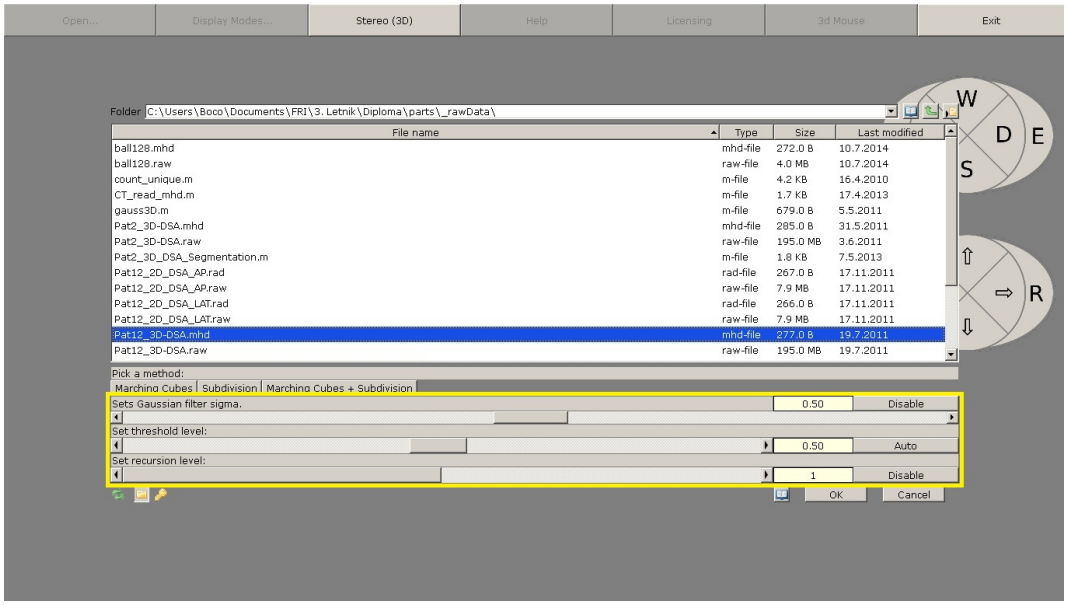
koliko globoko naj seže rekurzija, oziroma kako podrobno želimo izrisati naš model (Slika 4.6). Z vsako delitvijo na manjše kocke je delovanje algoritma, zaradi večjega števila podatkov, bolj kompleksno in je njegov čas izvajanja daljši.



Slika 4.6: Delovanje algoritma s povečevanjem stopnje rekurzije. Na prvi sliki je stopnja rekurzije enaka 1, na drugi sliki je 2 in na tretji sliki je 3.

4.3 Implementacija in primeri uporabe

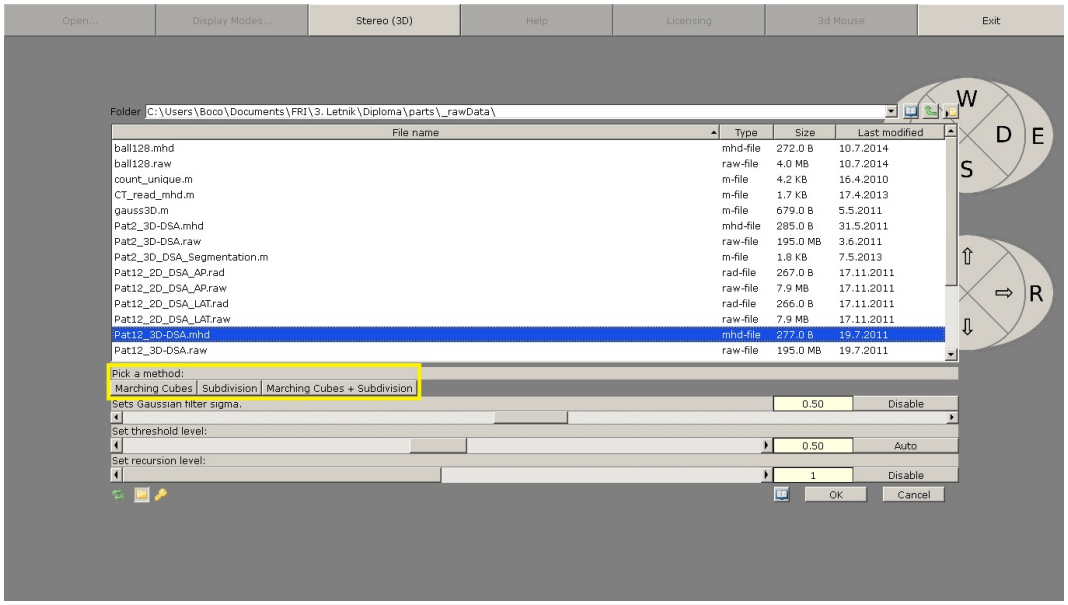
Po realizaciji algoritma je sledila implementacija v program NeckVeins in razširitev obstoječih algoritmov za segmentacijo z rekurzijo. Za namene diplomske naloge smo na grafični vmesnik dodali nove komponente (Slika 4.7). Novosti so drsnik, tekstovno polje in gumb za aktivacijo ali deaktivacijo rekurzivne metode. Z drsnikom se nastavlja globina, do katere bo rekurzija segla.



Slika 4.7: Z rumeno barvo so obkroženi drsniki za nastavljanje parametrov.

Pri algoritmih za segmentacijo je program NeckVeins že uporabljal Gaussovo metodo za glajenje in izrisovanje površine z metodo Marching Cubes. K tem metodam smo dodali še metodo za razbijanje v manjše kocke. Poleg tega smo implementirali tudi novo metodo, ki rekurzivno sega v globino in razbija kocke na manjše, vendar izrisuje površino na tak način, kot to počne metoda Marching Cubes. Uporabnik ima tako na razpolago 3 segmentacijske metode: Marching Cubes, rekurzivno deljenje na manjše kocke in Marching Cubes z

rekurzivnim deljenjem na manjše kocke. Za izbiro metode so bili v grafični vmesnik vgrajeni gumbi, s katerimi uporabnik potrdi svojo izbiro metode za segmentacijo (Slika 4.8).



Slika 4.8: Z rumeno barvo so obkroženi gumbi za izbiranje metode segmentacije.

V odvisnosti od izbrane metode se določene komponente zameglijo in njihovih vrednosti ni možno spreminjati. Tako se na primer pri izbiri metode za rekurzivno deljenje kock zamegli drsnik za spreminjanje sigma parametra Gaussovega filtra, ker se metoda ne uporablja.

Na svojo željo lahko uporabnik sam izbira in spreminja parametre, brez direktne izbire ene od metod. Vendar mora vedeti, da lahko na ta način izbira le med metodama rekurzivnega deljenja kock in Marching Cubes. Katero od dveh metod bo izbral, je pa odvisno od stanja omogočenosti/neomogočenosti, v katerem se nahaja drsnik za rekurzijo.

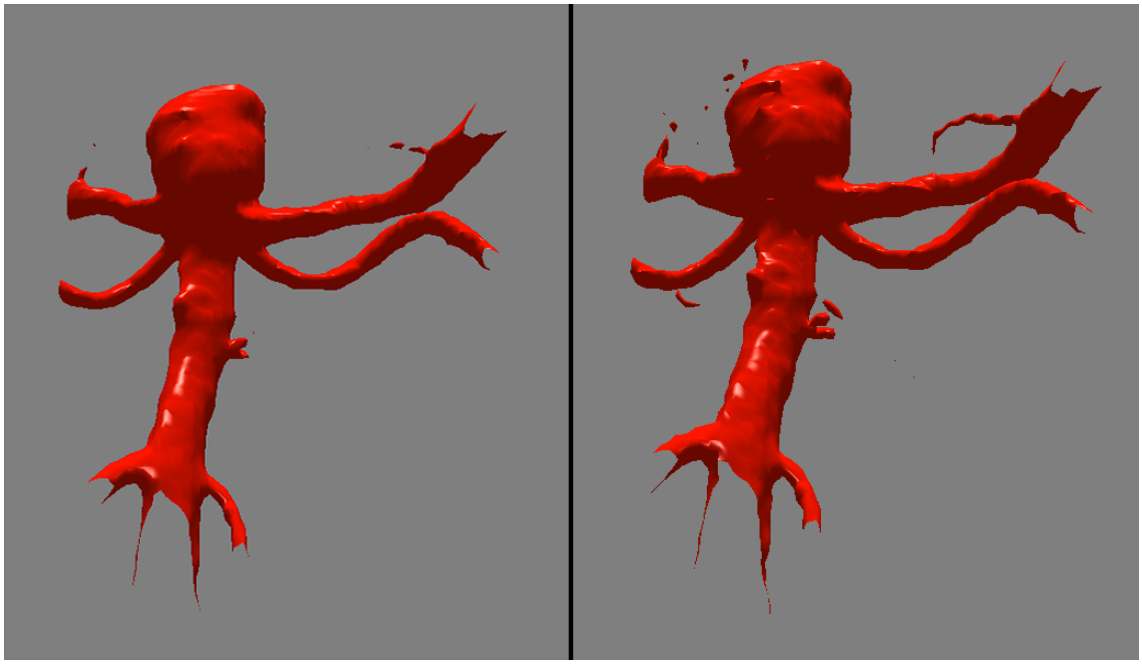
4.4 Primerjava segmentacijskih algoritmov

Za pridobitev informacije, ali je naša metoda lahko konkurenčna ostalim metodam za segmentacijo, moramo rezultate metod med seboj primerjati. V nadaljevanju so predstavljene zaslonske slike 3D modelov žil z uporabo posameznih izmed metod za segmentacijo. Vsaki od metod se lahko spreminja določene vrednosti parametrov, prav tako se lahko te parametre onemogoči.

Metode uporabljajo različne parametre, zato velja poudariti, katere parametre uporablja določena metoda:

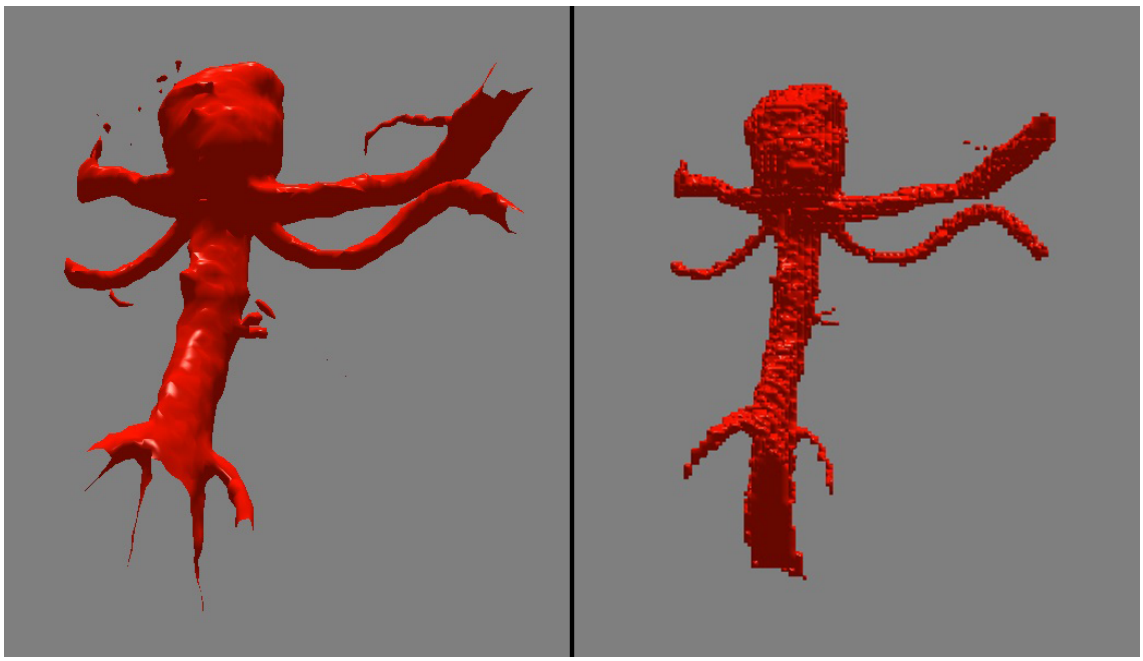
- Marching Cubes – Sigma parameter Gaussovega filtra, raven praga;
- Subdivision (rekurzivno deljenje kock) – raven praga, stopnja rekurzije;
- Marching Cubes in Subdivision - Sigma parameter Gaussovega filtra, raven praga, stopnja rekurzije.

Gaussov filter se v računalniški grafiki uporablja za glajenje slik in za odstranjevanje šuma. Pri primerjavi slik (Slika 4.9), na katerih je izrisan 3D model žile s pomočjo metode Marching Cubes, se nazorno vidi uporaba Gaussovega filtra. Na desni strani slike lahko vidimo prisotnost delčkov žile, ki jih glajenje odpravi.

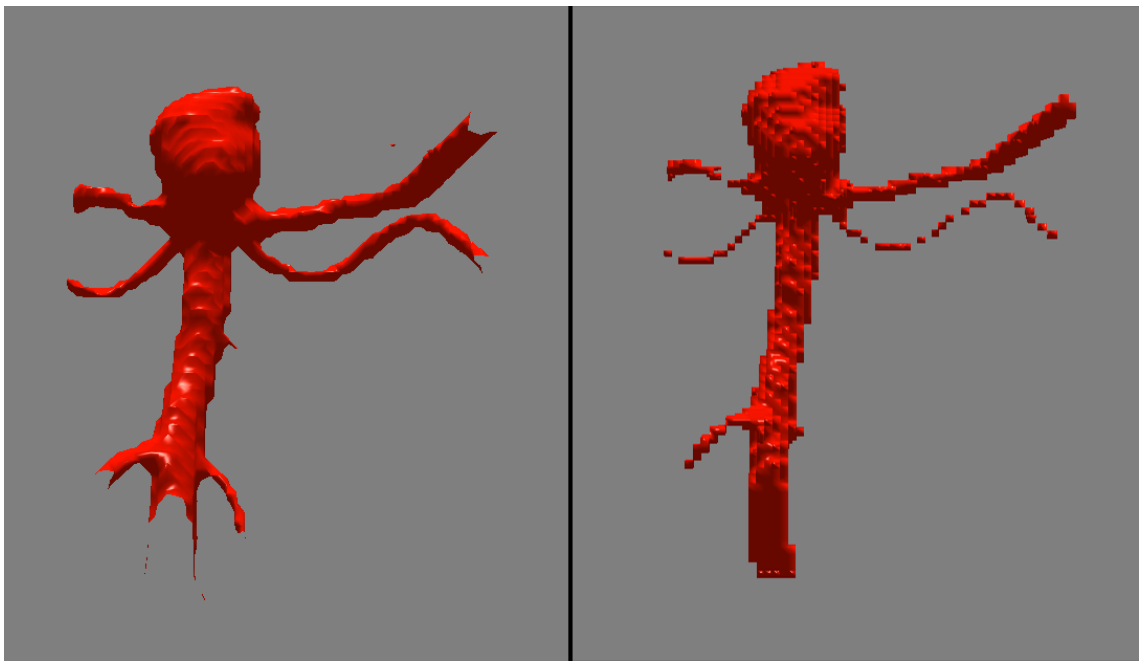


Slika 4.9: Primerjava izrisov metode Marching Cubes s Sigma parametrom (levo) in brez Sigma parametra (desno) Gaussovega filtra.

Če želimo podati splošno oceno naši razviti metodi, jo je potrebno primerjati z že obstoječimi. Sprva bomo metodo primerjali z algoritmom Marching Cubes (Slika 4.10) in (Slika 4.11), nato z različico algoritma Marching Cubes, v katerega smo implementirali rekurzivno deljenje vokslov.



Slika 4.10: Primerjava izrisov metode Marching Cubes (levo), brez Gaussovega filtra in mejo praga 0.6, in rekurzivnega deljenja kock (desno) s stopnjo rekurzije 2 in mejo praga 0.6.

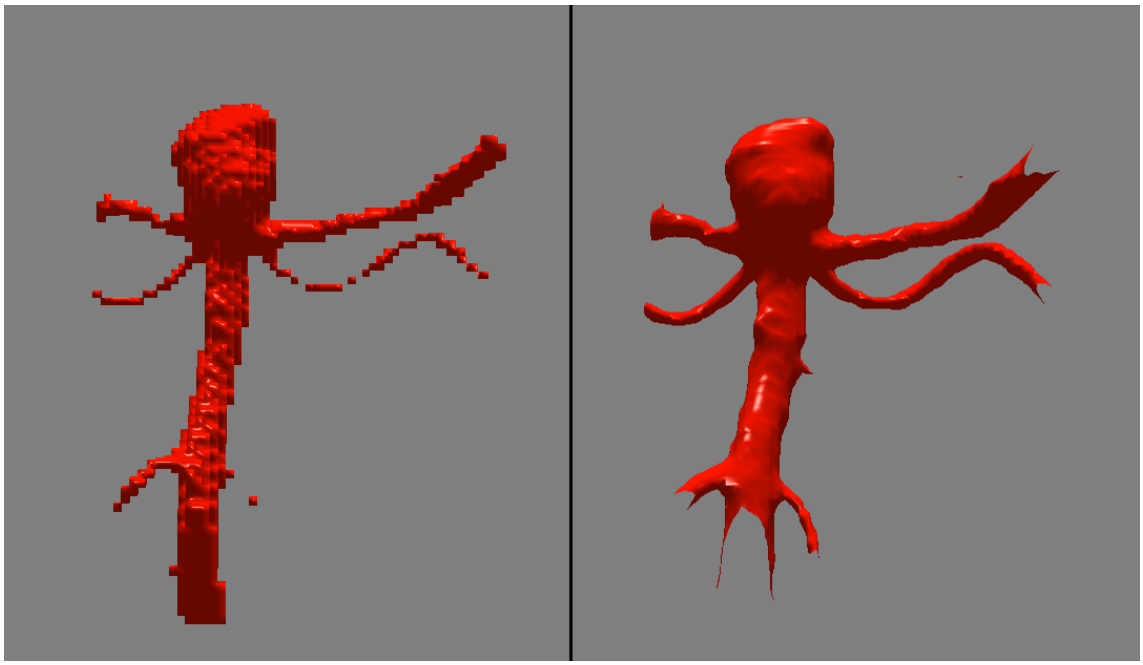


Slika 4.11: Primerjava izrisov metode Marching Cubes (levo), brez Gaussovega filtra in mejo praga 0.9, in rekurzivnega deljenja kock (desno) s stopnjo rekurzije 2 in mejo praga 0.9.

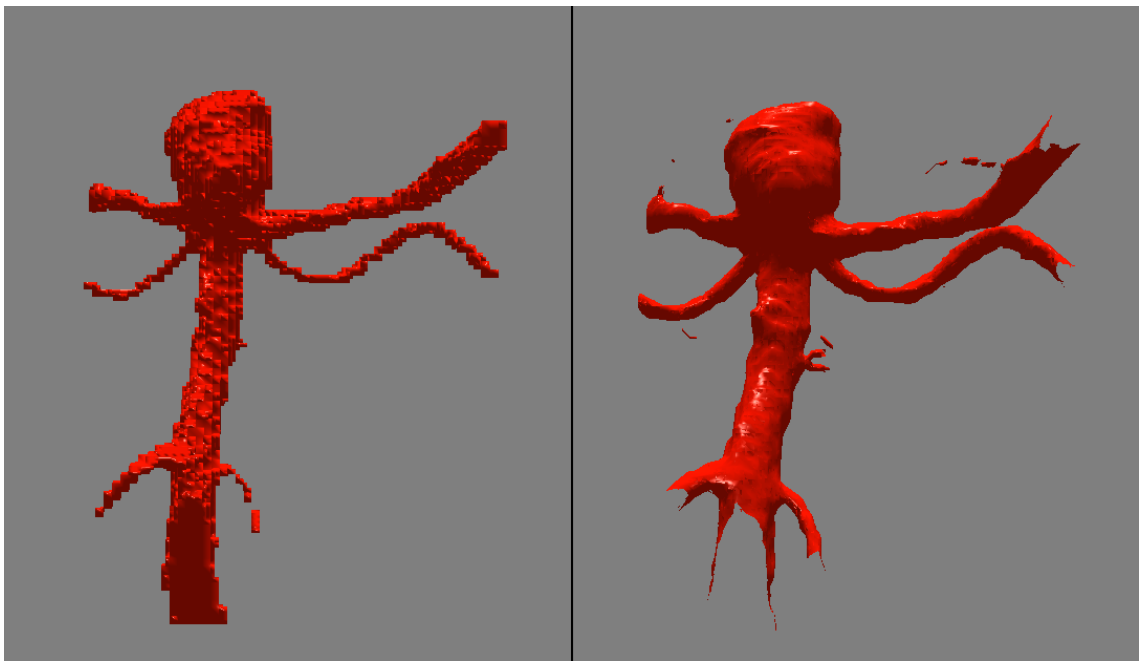
Na prvi sliki lahko vidimo, kako oba algoritma pri enakih parametrih izrišeta zelo podoben model. Prva podrobnost, ki jo lahko opazimo, je razlika v zglajenosti. Marching Cubes algoritem je namenjen rekonstrukciji površja, medtem ko je naš algoritem bolj osredotočen na gradnjo modela. Seveda bi manjšo stopničavost in s tem večjo zglajenost lahko dosegli z večjo globino rekurzije. Na drugi sliki smo povečali parameter meje praga. Več kot očitno je povečava parametra poslabšala izris modela, predvsem v primeru naše metode.

Različica metode Marching Cubes, z implementiranim rekurzivnim deljenjem vokslov, je bila ustvarjena za testiranje. Želeli smo videti, ali je možno rekurzivno metodo deljenja vokslov implementirati v druge algoritme in njeno delovanje primerjati z originalno metodo rekurzivnega deljenja (Slika 4.12) in

(Slika 4.13).



Slika 4.12: Primerjava Rekurzivnega deljenja kock (levo) in Marching Cubes z implementirano rekurzijo za deljenje kock (desno). Kot parametre smo vzeli nivo rekurzije 1 in mejo praga 0.7.



Slika 4.13: Primerjava Rekurzivnega deljenja kock (levo) in Marching Cubes z implementirano rekurzijo za deljenje kock (desno). Kot parametre smo vzeli nivo rekurzije 2 in mejo praga 0.7.

Različica Marching Cubes metode je vsekakor potrdila, da je naša rekurzivna metoda segmentacije integrabilna z že obstoječimi metodami. Pri primerjanju izrisanega modela je med njima vidna razlika, predvsem v stopničavosti in zglajenega površja. V različici Marching Cubes metode lahko opazimo, da stopnja rekurzije nima tako pomembne vloge, kot jo ima pri metodi z rekurzivnim deljenjem kock. Na splošno bi lahko rekli, da smo z delovanjem naše metode dosegli zadane cilje.

4.4.1 Časovna in prostorska kompleksnost

Za konec smo naredili še analizo časovne in prostorske kompleksnosti. Merjenja časov izvajanja so se izvajala na prenosnem računalniku z 64-bitnem opera-

cijskim sistemom Windows 7, Intel Core 2 Duo T6600 2.2GHz procesorjem in 4GB DDR3 rama. Pri časovni kompleksnosti smo naredili primerjavo hitrosti izvajanja algoritmov za segmentacijo z različnimi parametri. Kot čas izvajanja smo upoštevali samo dejanski čas izvajanja metode do vrnitve iz nje. Rezultati meritev so prikazani v tabelah 4.1, 4.2 in 4.3.

	$t(0.5)$			$t(0.7)$		
velikost podatkov	$r(1)$	$r(2)$	$r(3)$	$r(1)$	$r(2)$	$r(3)$
$30 \times 30 \times 30$	$0.358s$	$0.523s$	$0.641s$	$0.132s$	$0.989s$	$2.243s$
$60 \times 60 \times 60$	$2.568s$	$3.259s$	$3.344s$	$2.369s$	$4.074s$	$8.319s$
$80 \times 80 \times 80$	$39.199s$	$39.968s$	$40.649s$	$35.484s$	$24.635s$	$31.372s$

Tabela 4.1: Tabela prikazuje izmerjene čase za našo rekurzivno metodo. Med seboj smo primerjali metodo pri nižji $t(0.5)$ in višji $t(0.7)$ meji za mininalni prag. Prav tako smo za vsako stopnjo izvedli testiranja z različnimi stopnjami rekurzije (stopnje 1, 2 in 3).

	$t(0.5)$		$t(0.7)$	
velikost podatkov	$G(0.5)$	$G(disabled)$	$G(0.5)$	$G(disabled)$
$128 \times 128 \times 128$	$1.770s$	$0.553s$	$1.917s$	$0.934s$

Tabela 4.2: Primerjava Marching Cubes metode z različnji vrednosti parametra meje praga in sigma parametra Gaussovega filtra.

	$t(0.5)$			$t(0.7)$		
velikost podatkov	$r(1)$	$r(2)$	$r(3)$	$r(1)$	$r(2)$	$r(3)$
$30 \times 30 \times 30$	$0.037s$	$0.132s$	$0.653s$	$0.076s$	$5.425s$	$23.355s$
$60 \times 60 \times 60$	$0.243s$	$0.333s$	$0.816s$	$0.348s$	$8.496s$	$49.844s$
$80 \times 80 \times 80$	$0.386s$	$0.59s$	$0.958s$	$0.557s$	$9.901s$	$62.322s$

Tabela 4.3: Primerjali smo tudi delovanje različice metode Marching Cubes z implementiranim rekurzivnim deljenjem vokslov. Zabeležili smo čase izvajanja pri različnih vrednostih parametra za mejo praga in pri različnih stopnjah rekurzije.

Za prostorsko kompleksnost smo analizirali zgradbo našega algoritma in podali teoretično osnovo. Naš algoritem ima glavno metodo, v kateri se sprehodimo čez 3D matriko vokslov, torej imamo časovno kompleksnost reda a^3, kjer je a velikost vhodnih podatkov. Znotraj glavne metode se kliče rekurzivna metoda. Rekurzivna metoda ima v odvisnosti od izbrane globine rekurzije časovno kompleksnost reda b^n, kjer je b vedno enak 8 (za vsak voksel, ki ga želimo razdeliti na manjše voksle, se sprehodimo čez vsa oglišča), n pa je maksimalna vrednost globine. Osnovna formula kompleksnosti se tako glasi:

$$a^3 + b^n \tag{4.2}$$

Iz formule se da izpeljati še nekoliko več. Funkcija a^3 narašča veliko počasneje kot funkcija 8^n. Za velike vrednosti a in n bi lahko formulo posplošili kar v c^n, kjer je c konstanta. Za probleme, kot smo jih sami testirali in kjer je bil n tipično majhna številka, je tako veljala formula $a^3 + c$. Pri računanju kompleksnosti je tako skozi prisotna eksponentna funkcija, ki jasno pokaže, da je metoda požrešna.

Poglavje 5

Zaključek in sklepne ugotovitve

V nalogi smo implementirali metodo za segmentacijo z rekurzivnim deljenjem vokslov na manjše segmente. Za lažjo implementacijo v obstoječi program NeckVeins, smo metodo napisali v Javi. Sledila je še razširitev algoritma Marching Cubes z na novo ustvarjeno metodo z rekurzivnim deljenjem vokslov. Za konec smo izrisane modele segmentacijskih algoritmov primerjali med seboj.

Razvito metodo bi lahko v nadaljnjem še izboljšali. Zanimivo bi bilo implementirati način za prepoznavanje tankih žil oziroma predelov ožilja. Metodo bi lahko pohitrili tudi z drugačno obliko vhodnih podatkov, na primer tako, ki ne vsebuje nepotrebnih podatkov o vokslih, ki predstavljajo ozadje. Vsekakor bi bilo zanimivo videti rezultate pri višjih stopnjah rekurzije. Za to potrebujemo zmogljivejši računalnik, predvsem pa zmogljivejšo grafično kartico z zadostno količino pomnilnika.

Delovanje prototipa programa NeckVeins je bilo že predstavljeno zdravnikom na Univerzitetnem kliničnem centru Ljubljana. Odzivi zdravnikov so bili pozitivni in prototip je bil deležen kopice spodbudnih besed. V bodoče lahko pričakujemo še medsebojnega sodelovanja in testiranja aplikacije. Morda bo nekega dne projekt prišel do stopnje razvoja, ki bo ustrezala uporabi v zdra-

vstvu, in tako pomagal zdravnikom pri diagnosticiranju in zdravljenju.

Literatura

[1] LWJGL. http://www.lwjgl.org/, 2014.

[2] OpenGL. http://www.opengl.org/, 2014.

[3] Paul Bourke. Polygonising a scalar field. http://paulbourke.net/geometry/polygonise/, 1994.

[4] Maryam Taghizadeh Dehkordi, Saeed Sadri, in Alimohamad Doosthoseini. A review of coronary vessel segmentation algorithms. *Journal of medical signals and sensors*, 1(1):49, 2011.

[5] T Todd Elvins. A survey of algorithms for volume visualization. *ACM Siggraph Computer Graphics*, 26(3):194–201, 1992.

[6] Image Quest INC. Imaging service. Xray hand. http://4imagequest.com/ultrasound-repairs/, 2014.

[7] Cemil Kirbas in Francis Quek. A review of vessel extraction techniques and algorithms. *ACM Computing Surveys (CSUR)*, 36(2):81–121, 2004.

[8] Christoph Kubisch, Sylvia Glaßer, Mathias Neugebauer, in Bernhard Preim. Vessel visualization with volume rendering. Objavljeno v *Visualization in Medicine and Life Sciences II*, strani 109–132. Springer, 2012.

[9] David Lesage, Elsa D Angelini, Isabelle Bloch, in Gareth Funka-Lea. A review of 3D vessel lumen segmentation techniques: Models, features and extraction schemes. *Medical image analysis*, 13(6):819–845, 2009.

[10] Ken Museth, Torsten Möller, Anders Ynnerman, et al. Model-free surface visualization of vascular trees. 2008.

[11] Andreas Pommert in Karl Heinz Höhne. Validation of medical volume visualization: a literature review. Objavljeno v *International Congress Series*, del 1256, strani 571–576. Elsevier, 2003.

[12] Bernhard Preim in Steffen Oeltze. 3D visualization of vasculature: an overview. Objavljeno v *Visualization in medicine and life sciences*, strani 39–59. Springer, 2008.

[13] Darkhorse Productions. Slike v medicini. `http://www.medenosrce.net/component/attachments/download/5054`, 2006.

[14] Anže Sodja. *Segmentacija prostorskih medicinskih podatkov na GPE*. Doktorska disertacija, Fakulteta za računalništvo in informatiko, 2013.

[15] Simon Žagar. *Vizualizacija žil tilnika z OpenGL*. Doktorska disertacija, Fakulteta za računalništvo in informatiko, 2012.

[16] WHO. The top 10 causes of death. `http://www.who.int/mediacentre/factsheets/fs310/en/`, 2002.

[17] Wikipedia. List of causes of death by rate — Wikipedia, The Free Encyclopedia. `http://en.wikipedia.org/wiki/List_of_causes_of_death_by_rate`, 2010.

[18] Wikipedia. Eclipse (software) — Wikipedia, The Free Encyclopedia. `http://en.wikipedia.org/wiki/Eclipse_(software)`, 2014.

[19] Wikipedia. Gaussian blur — Wikipedia, The Free Encyclopedia. `http://en.wikipedia.org/wiki/Gaussian_blur`, 2014.

[20] Wikipedia. Java (programming language) — Wikipedia, The Free Encyclopedia. `http://en.wikipedia.org/wiki/Java_(programming_language)`, 2014.