VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REKONSTRUKCE 3D GEOMETRIE NA ZÁKLADĚ DISKRÉTNÍCH VOLUMETRICKÝCH DAT

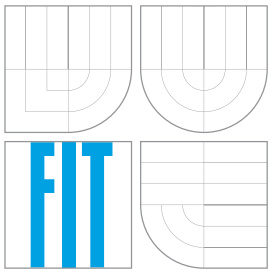
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

RADEK SVĚCHOVSKÝ

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REKONSTRUKCE 3D GEOMETRIE NA ZÁKLADĚ DISKRÉTNÍCH VOLUMETRICKÝCH DAT

3D GEOMETRY RECONSTRUCTION FROM DISCRETE VOLUMETRIC DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — RADEK SVĚCHOVSKÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL ŠPANĚL, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Převod diskrétních volumetrických dat na jejich povrchovou reprezentaci je dnes relativně běžnou operací. Standardním řešením tohoto problému je užití algoritmu Marching cubes, který ačkoli je jednoduchý a robustní, produkuje nekvalitní výstup, který vyžaduje následný post-procesing. Tato diplomová práce se zabývá studiem alternativních algoritmů pro extrakci izoploch z objemových dat. Čtenář bude srozuměn s fundamenty této problematiky a s principy metody Hierarchical Iso-Surface Extraction, jejíž nezávislá implementace byla v rámci této práce provedena a testována.

Abstract

Conversion of discrete volumetric data to boundary representation is quite common operation. Standard approach to resolve this problem is to use well-known Marching cubes algorithm, which although simple and robust, generates low-quality output that requires subsequent post-processing. This master's thesis deals with uncommon algorithms used for isosurface extraction from volumes. The reader will be acquainted with fundamental principles of Hierarchical Iso-Surface Extraction method, that was independently implemented and tested in this work.

Klíčová slova

objemový model, voxel, polygonální model, izoplocha, rekonstrukce, extrakce, Marching cubes

Keywords

volumetric model, voxel, polygonal model, isosurface, reconstruction, extraction, Marching cubes

Citace

Radek Svěchovský: Rekonstrukce 3D geometrie na základě
diskrétních volumetrických dat, diplomová práce, Brno, FIT VUT v Brně, 2013

Rekonstrukce 3D geometrie na základě diskrétních volumetrických dat

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Španěla, Ph.D. Uvedl jsem všechny literární zdroje a publikace, ze kterých jsem čerpal.

.....
Radek Svěchovský
22. května 2013

Poděkování

Rád bych tuto práci věnoval svým rodičům, kteří mě po celou dobu mého vysokoškolského studia podporovali ve chvílích úspěchu i nezdaru. Dále bych rád poděkoval svému současnému vedoucímu práce, panu Ing. Michalu Španělovi, paradoxně za určitou volnost v řešení, kterou mi poskytl a také za to, že byl ochoten vést toto téma, ačkoli jej neinicioval. Nemohu také opomenout svého předchozího vedoucího semestrálního projektu, pana docenta Přemysla Krška, jenž byl nejenom mým předešlým školitelem, ale také mi poskytl rady, které lze zužitkovat v budoucím životě. A v neposlední řadě patří můj dík všem, kteří implementují knihovny MDSTk, OpenMesh a ostatní nástroje, které jsem ve své práci používal a považoval za úplnou samozřejmost.

Obsah

Kapitola 1

Úvod

Extrakce a vizualizace izoploch je v dnešní době standardním postupem pro zobrazování a rekonstruování povrchu volumetrických dat. Jeden z prvních algoritmů, *Marching Cubes*, umožňující převod objemových dat na povrchovou reprezentaci se stal, z důvodu své robustnosti a jednoduchosti, také velmi rozšířeným a používaným postupem této oblasti vizualizací a to i přes jeho zjevné nevýhody. Kromě tohoto klasického přístupu je ale neustále snaha pokračovat ve výzkumu a hledat jiné cesty pro extrakci izoploch z volumetrických dat.

Tato diplomová práce je zaměřena na analýzu existujících alternativních přístupů zobrazování izoploch a na výběr a implementaci některého z těchto algoritmů. V rámci diplomové práce je implementován, testován, srovnáván a na základě výsledků testů a analýz i modifikován za účelem zrychlení výpočtu a zjednodušení implementace algoritmus emphHierarchical Iso-Surface Extraction.

Semestrální projekt v tomto případě posloužil pouze jako studie existujících extrakčních algoritmů.

V první kapitole této práce jsou zavedeny a vysvětleny pojmy nezbytně nutné pro výklad a popis algoritmu, jeho modifikací a testování. V kapitole druhé se nachází zevrubný popis existujících přístupů k extrakci a vizualizaci objemových dat, a dále následuje podrobný rozbor implementovaného algoritmu, zejména proto, že je prakticky složen z několika dalších nezávislých algoritmů, jejichž funkcionalita není triviální. Poté jsou uvedeny základní informace týkající se implementace následovány kapitolou o testování. Závěrečná kapitola obsahuje zhodnocení prací a nastiňuje možný budoucí rozvoj projektu.

Kapitola 2

Reprezentace objektů v trojrozměrném prostoru

Na reprezentaci trojrozměrných objektů jsou kladeny různorodé, za určitých okolností dokonce protichůdné, požadavky vyplývající z vlastností zdroje a výsledného použití dat. Zdrojem dat se v tomto případě může stát stroj (např. mračna bodů v prostoru vzniklá 3D skenováním laserovým nebo dotykovým (tactile) zařízením, data v uspořádané kartézské mřížce jako výstup magnetické rezonance (MRI) nebo počítačové tomografie (CT), data vzniklá simulací např. použitím Metody konečných prvků) nebo člověk (Constructive Solid Geometry (CSG) model strojního zařízení, polygonální model vzniklý pro potřeby herního či filmového průmyslu apod.). Výsledná data mohou být zobrazována (nezřídka v reálném čase) na zobrazovacích zařízeních, můžeme požadovat vyrobení takového modelu, např. na 3D tiskárnách nebo s využitím CNC zařízení nebo je možné data dále zpracovávat, případně převádět na jiné reprezentace. Kromě specifických požadavků daných doménou použití 3D modelu jsou zde požadavky obecné – co možná nejnižší paměťové nároky, jednoznačnost interpretace (což není samozřejmostí, viz drátový model), obecnost, nebo rychlost zpracování.

Proto přirozeně vzniklo značné množství datových struktur umožňujících uchování 3D objektů s ohledem na potřeby domény aplikace. Protože je tato práce zaměřena právě na konverzi mezi dvěma různými datovými reprezentacemi, objemovou a povrchovou, jsou v této kapitole ve zkratce nastíněny základní typy modelů právě se zaměřením na volumetrický a polygonální model. Dále zde čtenář nalezne důležité fundamenty použité v pozdějších kapitolách této publikace. Některé z obecných informací byly čerpány z knihy Moderní počítačová grafika [12].

2.1 Volumetrický model

V případě, že jsou k dispozici pouze vzorky dat na určitých souřadnicích v prostoru a nevyjadřují povrch tělesa (jako je tomu u bodové reprezentace, viz kapitolu 2.2), jedná se o *objemový model*. Takové modely mohou uchovávat jako vzorky data skalární i vektorová, často neobrazového charakteru – hustotu, míru pohlcení rentgenového záření, teplotu v daném bodě apod. Vzorky mohou být prostorově neuspořádané, v takovém případě se jedná o tzv. *rozptýlená data* a je zapotřebí uchovat jednoznačné souřadnice vzorku. Zdrojem takových dat jsou zpravidla různá měření s nezávisle rozmístěnými snímači – meteorologická měření, testování ve větrných tunelech a podobně. Druhou variantou je uspořádání

vzorků do mřížek. Ty mohou být nepravidelné (strukturované, blokově strukturované, nestrukturované či hybridní) získané např. simulací proudění, nebo pravidelně uspořádané (do kartézské, pravidelné nebo alespoň do pravoúhlé mřížky, která zpravidla vzniká převedením z jiného než kartézského souřadného systému – např. ze sférických či cylindrických souřadnic). Taková data bývají nejčastěji získána jako výstupy MR či CT snímků, kde jsou jednotlivé 2D řezy[1] seřazeny a spojeny. Více informací o typologickém rozdělení objemových dat lze nalézt v knize J. Žáry [12].

V rámci této práce je řešen převod pravidelně uspořádaných objemových dat, proto bude detailněji rozebrán tento typ volumetrických modelů. Spojitá data v n-rozměrném prostoru jsou z matematického pohledu *zobrazením*

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

nicméně vzhledem k tomu, že výpočetní technika je zpravidla diskrétní, musí být tato data navzorkována v určitých bodech. Nejčastěji jsou vzorky naměřeny v trojrozměrném prostoru, takový diskrétní trojrozměrný model je tedy zobrazením

$$f : \mathbb{G} \to \mathbb{R}^m$$

kde $\mathbb{G} = \{(x_i, y_j, z_k) : 0 \leq i \leq n_x, 0 \leq j \leq n_y, 0 \leq k \leq n_z, \}$, kde n_x, n_y, n_z jsou rozměry mřížky v dimenzích x, y a z. Pokud jsou vzorky uspořádány do pravidelné mřížky, pak platí, že $x_i = x_0 + i \cdot h_x$, $y_j = y_0 + j \cdot h_y$ a $z_k = z_0 + k \cdot h_z$. Odtud je jasně zřejmé, že v takovém případě postačí uchovat jednotkové rozměry mřížky h_x, h_y, h_z a počáteční souřadnice[2]. Bez újmy na obecnosti bude dále předpokládáno, že $h_x = h_y = h_z$ a tedy, že se jedná o kartézskou pravidelnou mřížku. Zároveň lze bez újmy na obecnosti očekávat vzorkování skalárních dat a tedy

$$f : \mathbb{G} \to \mathbb{R}$$

Navíc, vzhledem ke konečné přesnosti datových typů jsou vzorky kvantovány a odtud tedy vychází matematický popis diskrétních volumetrických dat uspořádaných do kubické mřížky

$$f : \mathbb{G} \to \mathbb{N} \tag{2.1}$$

podobně, jako je zaveden v materiálu U. Labsika a kolektivu [2].

Jeden vzorek objemového modelu se nazývá *voxel* (název vznikl jako analogie k 2D grafickému elementu – pixelu). Jeho rozměr je dán jako h_x, h_y, h_z, jedná se tedy o kvádr, příp. krychli a všechny vzorkované veličiny mají v daném voxelu konstantní hodnotu. To znamená, že hodnoty v místech neodpovídajících žádnému z vzorků jsou interpolovány řádem 0 – nejbližším sousedem. Toto může být v určitých případech omezující a proto některé algoritmy interpolují hodnotu v daném místě emphtrilineárně na základě tzv. *buňky*[3]. Buňka je tvořena 8 incidentními vzorky na souřadnicích $x \in \{x_i, x_{i+1}\}, y \in \{y_j, y_{j+1}\}, z \in \{z_k, z_{k+1}\}$.

Dalším velmi důležitým pojmem je *izoplocha*[4]. Jedná se o povrch procházející místem objemu s určitou konstantní hodnotou – *izohodnotou*[5] – vzorkované veličiny. Obecně je to tedy množina bodů

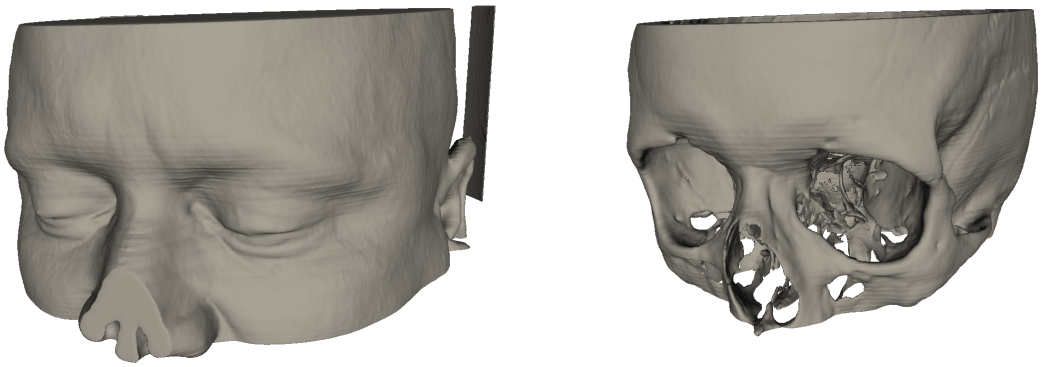
$$S(v) = \{x \in \mathbb{R}^n : f(x) = v\}$$

[1] Angl. slice
[2] I tato informace nemusí být uchována v případě, že absolutní souřadnice nejsou relevantní.
[3] Z angl. cell
[4] Z angl. isosurface
[5] Volně přeloženo z anglického isovalue

v případě kvantovaných dat ve 3D tedy

$$S(v) = \{(x, y, z) : \widetilde{f}(x, y, z) = v\} \tag{2.2}$$

kde $\widetilde{f} : [\mathbb{G}] \to \mathbb{R}$ je spojité rozšíření zobrazení f 2.1, které trilineárně interpoluje hodnoty $f(\mathbb{G})$, z toho důvodu, aby bylo možno zobrazit izoplochy i jiné izohodnoty, než jaká se vyskytuje v místech vzorkování. Na obrázku 2.1 jsou vidět dvě rozdílné izoplochy nad stejným objemovým modelem. Vlevo je izohodnota rovna `-750`, vpravo pak `520`.



Obrázek 2.1: Zobrazení rozdílných izoploch stejného objemu

2.2 Polygonální model

Výstupním datovým typem extrakce isoplochy z objemového modelu je povrchová reprezentace. Polygonální model je speciálním případem hraniční reprezentace [6] těles, do které spadají také bodová mračna, drátové modely a spline modely. Takové modely jsou definovány pouze svým povrchem, informace o vnitřní struktuře tělesa, nebo jiné, než grafické informace se zpravidla také neuchovávají.

Modely popsané hraniční reprezentací lze rozdělit z hlediska jejich reálné vyrobitelnosti. Fyzicky vyrobitelné modely se nazývají *manifold modely* (také *2-manifold*). Modely obsahující např. bodové spoje, nebo nekonečně tenké hrany, které jsou pochopitelně nevyrobitelné, se nazývají *non-manifoldy*.

Polygonální model je nestrukturovaná, po částech lineární a nespojitá síť nejčastěji tvořená trojúhelníkovými primitivy. Zásadní výhoda využití trojúhelníků jakožto základního primitiva oproti mnohoúhelníkům vyššího řádu je zřejmá – trojúhelník je vždy konvexní a jeho vrcholy náleží jedné rovině. Lze jej také poměrně jednoduše rasterizovat, a proto je možné i poměrně velké polygonální modely (skládající se z jednotek milionů primitiv) zobrazovat v reálném čase na grafických akcelerátorech, kde je převod vektorového popisu na diskrétní matici pixelů implementován v hardwarové rasterizační jednotce.

Základními jednotkami polygonálních modelů jsou vrcholy (také vertexy), hrany a plochy. Obecný polygonální model je matematicky n-tice (K, f), kde $K \subseteq 2^V$, kde f je zobrazení $V \to \mathbb{R}^3$ a V je neprázdná množina vrcholů. $E = \{e \in K : |e| = 2\}$ je potom množina hran a $F = \{f \in K : |f| = 3\}$ je množina ploch. Z hlediska vyrobitelnosti je u polygonálních

[6] Anglicky Boundary Representation, zkráceně pak také B-Rep

modelů manifold definován jako model, který neobsahuje volné vrcholy, každá z jeho hran sousedí právě se dvěma trojúhelníky a žádné dva trojúhelníky se vzájemně neprotínají.

Kromě geometrických informací (souřadnice vrcholů) je potřeba uchovávat i informace topologické. V zásadě nejjednodušším řešením je taková struktura, kdy každá hrana je schopna jednoznačně určit své dva a každá plocha své tři koncové vrcholy. Toto je úsporné z hlediska paměťových nároků a plně postačující z hlediska vykreslování grafickém hardware (kde je pouze potřeba určit, které 3 vrcholy tvoří příslušný trojúhelník), ale v případě složitějších algoritmů, kde je např. potřeba provádět výpočet nad okolím prvního řádu, je získání těchto informací výpočetně náročné.

Z těchto důvodů se v praxi používá datová struktura známá jako *okřídlená hrana*, kde každý záznam o hraně obsahuje informace o svých koncových vrcholech, incidentních hranách a o dvou přílehlých plochách. Nelze ji však využít pro popis nonmanifold tělesa. Existuje však datová struktura zvaná *půlhrana*, kde je každá hrana rozdělena na dvě směrově orientované půlhrany obsahující informace o svém vrcholu, o své ploše, o následující půlhraně, o protější půlhraně (pokud existuje) a případně o předchozí půlhraně. Tato struktura, kromě toho, že díky orientovaným hranám je ve vyhledávání určitých topologických informací rychlejší, navíc může uchovávat určité nonmanifold objekty. Hrana, která je incidentní pouze s jednou plochou se pak nazývá *hraniční*. Vedle přidaných topologických informací pak takové struktury mohou nést ještě další geometrická data, např. normály, případně i další informace, například stupeň plochy trojúhelníku v případě tzv. *balanced meshes* 4.7.

2.3 Ostatní reprezentace 3D objektů

2.3.1 Implicitní plochy

Mezi další neméně významné reprezentace patří *implicitní plochy* (též známé jako *Blobby objects* či *Meta balls*). Základem takového modelu je kostra tvořená generátory. Každý generátor má přiřazenu potenciálovou funkci, jejíž vyhodnocením pro danou izohodnotu získáme generovaný povrch. Tyto modely nabízejí vysokou úroveň abstrakce a zároveň malé paměťové nároky, nevýhodou je pak zobrazení. To může být provedeno přímo pomocí metod založených na sledování paprsku, kde dochází k analytickému vyhodnocování potenciálových funkcí nebo nepřímo převodem na polygonální model.

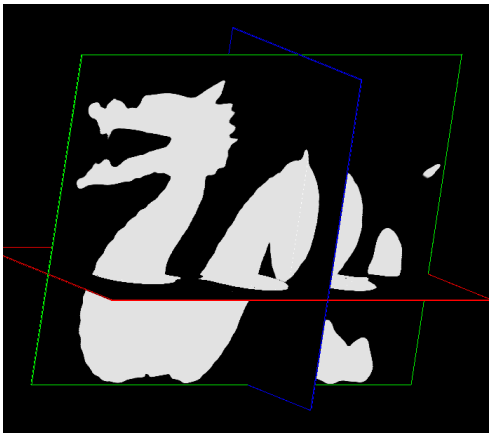
2.3.2 Constructive solid geometry

Konstruktivní geometrie těles je naopak založena na uspořádání jednoduchých těles (krychle, koule apod.) do *CSG* stromu a jejich kombinování pomocí množinových operací. Listy CSG stromu obsahují nezbytné informace o základních primitivech, vnitřní uzly pak množinové operace. Výhodou takové 3D reprezentace je především podobnost modelování se skutečnou výrobou objektů a zpravidla se používá v CAD systémech, nicméně převod takového tělesa na polygonální reprezentaci a jeho následné zobrazení není triviální.

Kapitola 3

Současný stav extrakce geometrie z volumetrických dat

Jak bylo naznačeno v předchozí kapitole, zobrazování volumetrických dat není jednoduchou intuitivní záležitostí. Objemová data lze sice zobrazovat přímo pomocí řezů rovnoběžných s rovinami souřadného systému, nicméně tato metoda zobrazení je značně limitující (viz obrázek 3.1). Extrakce izoploch je proto v dnešní době standardním prostředkem pro zobrazování 3D diskrétních volumetrických dat. Nejprve bude rozebrán současný stav poznání extrahování polygonálních modelů a pak následně bude částečně nastíněn postup přímého zobrazení volumetrických dat, jakožto konkurenta nepřímých zobrazovacích metod (tzv. *Indirect volume rendering* – IRD).



Obrázek 3.1: Zobrazení objemových dat pomocí řezů

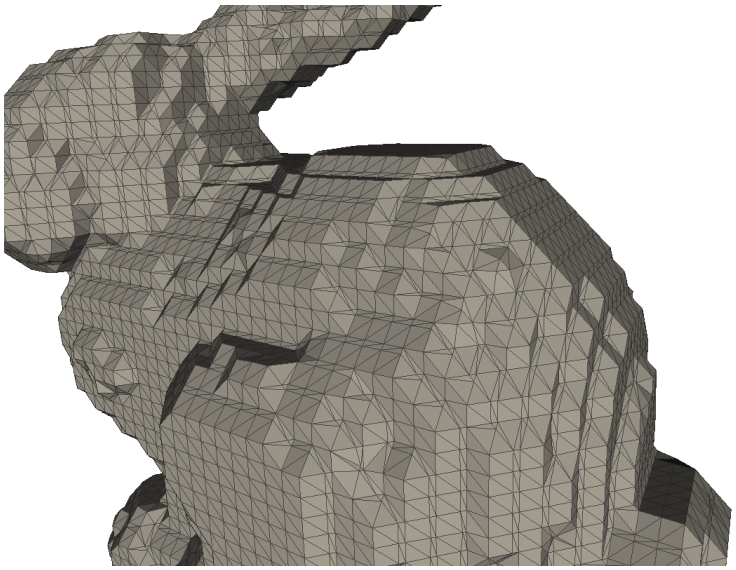
3.1 Nepřímé zobrazovací metody

Nepřímé metody zobrazení objemových dat převádějí určitou izoplochu na polygonální model, který je následně vyobrazen. Většina těchto technik bohužel zatím není schopna pracovat v reálném čase (snad s výjimkou implementací využívajících *GPGPU*, nicméně tyto implementace mají také svá omezení, např. po extrakci nelze modely ukládat a tím jsou značně eliminovány výhody nepřímého zobrazení), avšak po samotné extrakci trvající

řádově jednotky až desítky sekund je k dispozici polygonální model, který již je možno zobrazovat v reálném čase, lze jej libovolně zpracovávat, transformovat a ukládat.

3.1.1 Marching Cubes (s následným post-processingem)

Patrně nejrozšířenější metodou extrakce 3D geometrie z objemových modelů je algoritmus *Marching Cubes* (MC) [4]. Ten, ačkoli byl vymyšlen už v roce 1987, se stále těší velké oblibě a je hojně využíván. Postup vytvoření triangulace je prostý – procházením objemu buňku po buňce je postupně vytvářena triangulace na základě triangulační tabulky. Algoritmus má ale kromě svých nesporných výhod – jednoduchosti a robustnosti – také své nedostatky a proto je nutné výstupní polygonální modely zpravidla následně vyhladit (kvůli přítomnosti specifického šumu daného procesem extrakce, viz obrázek 3.2) a zdecimovat[1], případně je aplikován *remeshing*. Aplikace MC a následný post-processing je tak první skupinou postupů extrakce izoploch z volumetrických dat. Protože je algoritmus Marching Cubes dílčí součástí postupu *Hierachical Isosurface Extraction* [2] implementovaného v této práci, je detailněji popsán v kapitole 4.3.



Obrázek 3.2: Polygonální model jako výstup algoritmu Marching Cubes

3.1.2 Marching Tetrahedra (s následným post-processingem)

Jednou z nevýhod algoritmu MC v jeho základní podobě je také možnost vzniku nonmanifold modelů. Tomu předchází algoritmus *Marching Tetrahedra*, kde jsou vrcholy výsledného modelu umístěny na hrany čtyřstěnů, na které je každá buňka 8 voxelů rozdělena. Tímto nedochází k ne-jednoznačným výpočtům triangulace, nicméně tento algoritmu vytváří ještě větší množství trojúhelníků na buňku, než MC, u kterého lze považovat počet primitiv za neúčelný. Proto tento algoritmus nebývá příliš často využíván.

[1] Decimace je postup redukce počtu trojúhelníkových primitiv při zachování přesnosti popisu povrchu tělesa

3.1.3 Adaptivní shlukování voxelů

Další skupina algoritmů se snaží předejít generování nadbytečných primitiv spojováním voxelů. Jako zástupce této skupiny byl v rámci semestrálního projektu nastudován algoritmus *Multiresolution Isosurface Extraction with Adaptive Skeleton Climbing* [8]. Jádrem tohoto přístupu je slučování voxelů do bloků, ve kterých prochází zvolená izoplocha pouze jednou. Tento proces je proveden v 1D, 2D a následně ve 3D, čímž vzniknou maximální bloky voxelů, které je možno přímo triangulovat. Nicméně, jak sami autoři přiznávají, kvalita takto vytvořených modelů není nijak valná, povrch obsahuje značně degenerovaná primitiva a v případě jemnějšího nastavení se zase blíží výstupu algoritmu MC.

3.1.4 Hybridní metody

Do této skupiny metod extrakce povrchu z objemu lze zařadit různé postupy, které vhodně kombinují jednoduchý extrakční algoritmus, nejčastěji Marching Cubes, k získání základního hrubého povrchu, který je následně zjemňován a promítán na požadovanou izoplochu s pomocí remeshovacích algoritmů apod. Zástupcem této kategorie je *Multiresolution Isosurface Extraction with Adaptive Skeleton Climbing* [8], jemuž je, vzhledem k tomu, že byl implementován, věnována zvláštní kapitola 4.

3.1.5 Přímý výpočet triangulace

Poslední možností, jak získat polygonální model izoplochy, je pokusit se o přímou triangulaci. Takový postup je popsán v článku *High-Quality Extraction of Isosurfaces from Regular and Irregular Grids* [9], případně v *Semi-Regular Meshes Extraction from Volumes* [11]. Nejprve dojde k interpolaci, nebo aproximaci (v případě přítomnosti šumu) izoplochy pomocí splinů a následně je pak možné pro každý bod vypočítat zakřivení povrchu. Významné body s nejvyšší křivostí pro určitá okolí jsou pak řídícími body, na jejichž základě se počítá ideální délka hran nově vzniknuvších trojúhelníkových primitiv. Přidáváním nových trojúhelníků na okraj již triangulované části povrchu tak dochází k postupné triangulaci izoplochy.

3.2 Přímé zobrazovací metody

Direct Volume Rendering (DVR) metody slouží pouze k bezprostřednímu zobrazení objemu, jejich výstupem je tedy jen vizualizace dané izoplochy. Pro každý pixel výsledného obrazu je objemem puštěn paprsek, který jej v určitých místech navzorkuje (s využitím trilineární interpolace) a z těchto vzorků je pak postupně počítán barevný přírůstek pro daný pixel. Tato metoda, může být, vzhledem ke své vysoce paralelní povaze, snadno implementována pomocí GPGPU.

Kapitola 4

Hierarchical Iso-Surface Extraction

Tento algoritmus byl na základě srovnání v semestrálním projektu vybrán k implementaci, především proto, že poskytuje velice kvalitní výsledky, je modulární a lze tedy testovat různá nastavení a konfigurace sub-algoritmů a skýtá se zde určitý prostor pro vylepšení. Navíc, vzhledem k jeho hierarchické struktuře je možno generovat povrchy s různou přesností a úrovní detailů, což je možné aplikovat v jiných procesech jako je například komprese nebo progresivní přenos. V této kapitole bude Hierarchical Iso-Surface Extraction, stejně jako jeho jednotlivé dílčí algoritmy, podrobně rozebrán.

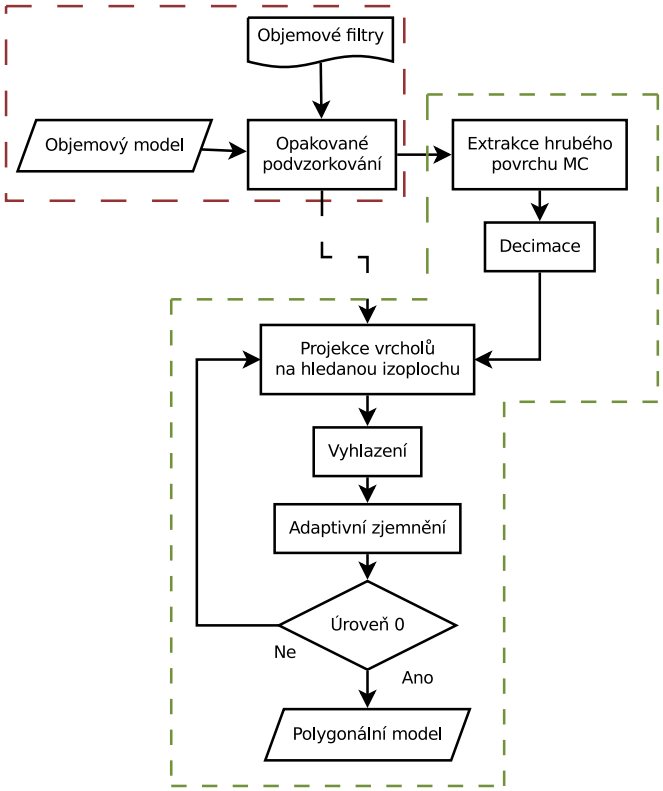
4.1 Obecný proces výpočtu triangulace

Postup extrakce lze v tomto případě rozdělit na část zpracovávající objemový model (červený oddíl na obr. 4.1) a na část generující a zpracovávající polygonální reprezentaci extrahované izoplochy (viz obrázek 4.1, zelený oddíl). V objemové části dochází k vytvoření hierarchické struktury volumetrických modelů, které jsou postupně podvzorkovávány vybraným objemovým filtrem. Nejvíce podvzorkovaný model obsahuje oproti základnímu objemu, vzhledem ke kvadratické závislosti, velmi malé množství voxelů. Zde dochází k extrakci izoplochy klasickým algoritmem Marching Cubes, ale vzhledem k tomu, že počet trojúhelníků je přímo úměrný počtu voxelů, kterými prochází izoplocha, je generované množství primitiv, při dostatečné velikosti hierarchie, snesitelné.

Následně je na takto vygenerovaný model aplikována jednoduchá decimace, která odstraňuje primitiva se špatným poměrem stran, která navíc kvůli svým drobným rozměrům nepřinášejí podstatnou geometrickou informaci. Zdecimovaný polygonální model je posléze připraven na postupné promítání na isosurface v objemu s vyšším stupněm detailu, až je nakonec promítnut na izoplochu původního volumetrického modelu. Po každém takovém promítnutí je potřeba aplikovat relaxační operátor, neboť zpravidla dochází k shlukování vrcholů a drobným průnikům trojúhelníků.

Také je samozřejmě zapotřebí zjemnit síť v místech, kde jsou původní primitiva příliš velká a nedostatečně interpolují zakřivení izoplochy. Zde je tedy aplikováno adaptivní dělení trojúhelníku a všechny nově vzniklé vrcholy jsou opět promítnuty na extrahovanou izoplochu. Takto vygenerovaný mesh je adaptivní, jemný v místech s vysokou křivostí povrchu, naproti tomu na rovných plochách jsou zachována velká primitiva a tak je celkový počet trojúhelníků přijatelný. Výstupní polygonální model nevyžaduje žádný další post-processing.

Výše zmíněný přístup se na první pohled příliš neliší od klasického postupu aplikování



Obrázek 4.1: Proces výpočtu Hierarchical Iso-Surface Extraction

MC, vyhlazovacího algoritmu a následné decimace, zvláště když všechny tři tyto kroky jsou zahrnuty v procesu výpočtu Hierarchical Iso-Surface Extraction. Přesto jsou zde dva velice důležité rozdíly. Jednak nedochází k vyrobení nesmyslně velkého počtu trojúhelníků, které pak jsou stejně zahozeny, ale naopak dochází k adaptivnímu dělení pouze v místech, která to vyžadují, a jednak je vše integrováno do jednoho celku a po celou dobu zpracování a zjemňování modelu je k dispozici objemová reprezentace. Ta je využívána k promítání a výpočtu potřeby rozdělení trojúhelníkových primitiv. To je klíčová odlišnost oproti klasickému vyhlazování a decimaci výstupu MC, protože zde již objemová data nefigurují a vše je tak odhadováno z původní triangulace. V následujících kapitolách budou podrobněji popsány jednotlivé kroky a výběr dílčích algoritmů.

4.2 Hierarchie objemových těles

Za účelem dosažení malého počtu primitiv na výstupu algoritmu Marching Cubes je potřeba původní objemový model podvzorkovat. Protože počet trojúhelníků kvadraticky závisí na velikosti hrany objemového modelu i pouhé podvzorkování v řádu jednotek vede ke znatelnému snížení počtu generovaných primitiv. Pokud je objem dán jako $f^0 : \mathbb{G}^0 \to \mathbb{N}$ 2.1, výsledná hierarchie je pak definována následující posloupností

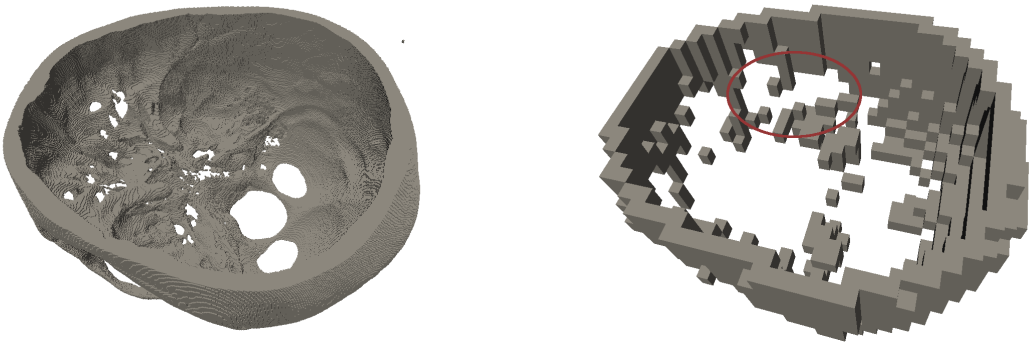
$$f^0, f^1, f^2, \ldots f^n$$

kde $f^l : \mathbb{G}^l \to \mathbb{N}$ pro $h_l = 2^l \cdot h_0$ a kde dimenze je $\lceil 2^{-l} n_x \rceil, \lceil 2^{-l} n_y \rceil$ a $\lceil 2^{-l} n_z \rceil$. Toho je dosaženo opakovanou konvolucí původního volumetrického modelu s určitým objemovým

filtrem. Mezi testované filtry patří *mean filtr*, *mediánový filtr* a *dilatační operátor*. Mean filtr vrací jako hodnotu nového voxelu střední hodnotu z odpovídající buňky. Lze jej definovat jako

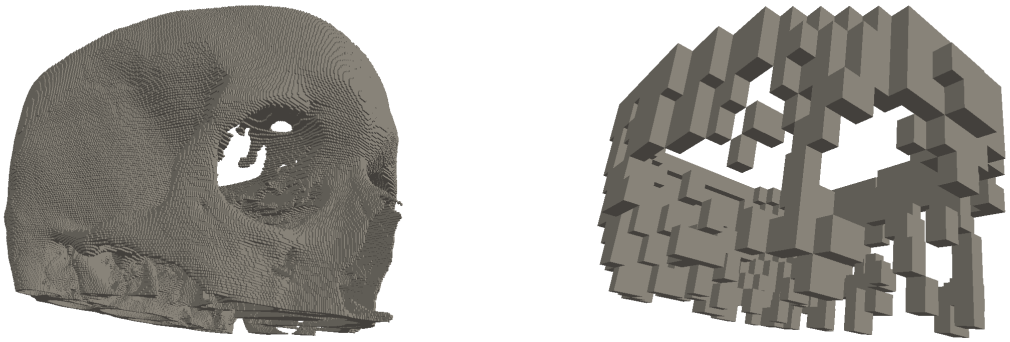
$$f^l(x_i, y_j, z_k) = \frac{1}{\mathrm{n}} \cdot \sum_{\substack{i' \in \{i,i+1\} \\ j' \in \{j,j+1\} \\ k' \in \{k,k+1\}}} f^{l-1}(x_{i'}, y_{j'}, z_{k'}) \tag{4.1}$$

kde l je úroveň hierarchie, $\mathrm{n} = 8$ (z definice buňky, viz stranu 4) a i, j, k jsou násobky 2^l. Mediánový filtr vzorky náležící množině $f^{l-1}(x_{i'}, y_{j'}, z_{k'})$, pro $i' \in \{i, i+1\}, j' \in \{j, j+$



Obrázek 4.2: Mean filtr: Vlevo původní objem, vpravo po 4 iteracích

$1\}, k' \in \{k, k+1\}$, kde l je opět úroveň hierarchie a i, j, k jsou taktéž násobky 2^l, nejprve seřadí a následně vybere hodnotu ze středu této číselné řady. Vzhledem k tomu, že vzorků je sudý počet, výsledkem je střední hodnota dvou vzorků. Oba tyto filtry se chovají ob-



Obrázek 4.3: Mediánový filtr: Vlevo původní objem, vpravo po 4 iteracích

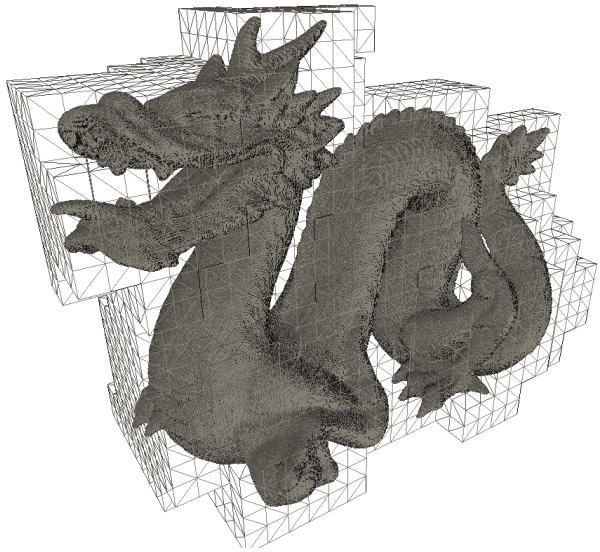
dobně a, na což autoři sami poukazují, mohou vytvářet díry v podvzorkovaných objemech. U objemných dat (navíc v případě, že je mezi hodnotami vzorků patřících extrahovanému tělesu a okolnímu prostředí velký rozdíl) k tomuto jevu nedochází, nicméně na reálných

datech obsahujících šum, drobné a tenké objekty opravdu vznikají nežádoucí díry, viz obrázek 4.2. Protože algoritmus není schopen jakýmkoli způsobem takto vzniklé díry zacelit, je tato vlastnost velmi nevyhovující a při použití vede v nejlepším případě k vytvoření non-manifoldu, kde se v místě původní díry vzájemně protnou plochy povrchu. Zároveň není možné později vytvořit triangulaci pro separátní objekt, který se vlivem podvzorkování na nejvyšším stupni hierarchie nenachází a tudíž není zpracován algoritmem Marching Cubes což vede k nenávratné ztrátě těchto částí modelu.

Řešením tohoto problému je podle článku [2] použití dilatačního operátoru pro podvzorkování objemu. Je definován jako

$$f^l(x_i, y_j, z_k) = \max_{\substack{i' \in \{i, i+1\} \\ j' \in \{j, j+1\} \\ k' \in \{k, k+1\}}} f^{l-1}(x_{i'}, y_{j'}, z_{k'}) \tag{4.2}$$

kde l, i, j a k jsou definovány výše. Tento filtr se chová opačně než předchozí dva zmíněné filtry a postupně zvětšuje původní volumetrický model. Pro každý f^l pak lze dokázat, že obsahuje f^{l-1}. Toto chování je vhodné pro topologicky jednoduché modely, nicméně má také

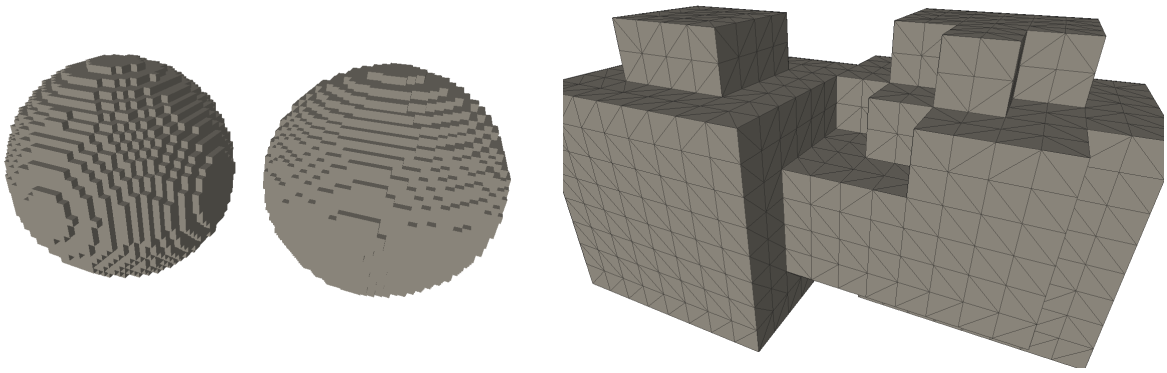


Obrázek 4.4: Dilatační filtr: Vnější obal je po 6 iteracích

svá omezení. Vlivem postupného růstu objemu může dojít k pohlcení a sjednocení blízkých objektů, které však opět v pozdějších fázích výpočtu není možno nijak rozdělit a proto výsledný model bude obsahovat určitá spojení mezi původně oddělenými izoplochami. Teoretické řešení tohoto problému je nastíněno zde 5.2. Po vytvoření hierarchie volumetrických těles následuje fáze extrakce izoplochy, která je provedena klasickým algoritmem Marching Cubes.

4.3 Marching Cubes

Ačkoli není algoritmus Marching Cubes z roku 1987 autorů W. E. Lorensena a H. E. Clina [4], prvním postupem pro extrakci izoploch, je patrně nejpoužívanější a dodnes je aplikován v široké škále software. Jedná se o sekvenční algoritmus procházející objem buňku po

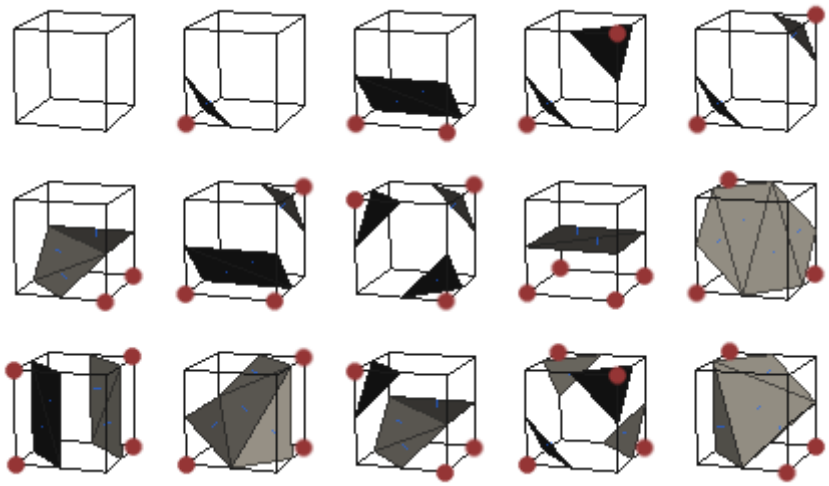


Obrázek 4.5: Splynutí objektů: Vlevo původní objem, vpravo po 3 iteracích dilatace

buňce, kde na základě předem připravené triangulační tabulky vytváří požadovanou izoplochu. Pro každou buňku čítající 8 voxelů nejprve dojde k prahování hodnot. V každém vrcholu tedy je tedy jednobitová informace v závislosti na tom, zda je daný vzorek větší nebo roven prahu (1) nebo zda je menší (0). Možných kombinací je přirozeně $2^8 = 256$. V případě, kdy všechny vzorky náleží po prahování stejné třídě ekvivalence zobrazení (to zn. že všechny jsou současně vyhodnoceny jako 0 nebo jako 1), tak je tato buňka zahozena a dále není zpracovávána, protože se celá nachází vně, nebo uvnitř extrahovaného objektu. V opačném případě je zřejmé, že takovou buňkou prochází vyhledávaná izoplocha alespoň jednou[1]. Posloupnost bitů tvořená výsledkem prahování pro jednotlivé vrcholy buňky tvoří index do tabulky triangulace, ze které je použita topologie primitiv. Poněvadž je krychle symetrické těleso, nebylo nezbytně nutné vytvořit všech 256 možných kombinací vnitřní triangulace. Původní tabulka (na obrázku 4.6) autorů obsahovala 15 konfigurací a využívala *rotační symetrii* a *reflexivitu*. Po zveřejnění se následně ukázalo, že za určitých okolností takto vytvořený polygonální model obsahuje díry a průniky hran v triangulaci, a tudíž je výsledný model nonmanifold, přestože by se za každých okolností mělo jednat o manifold. Tato chyba v původním algoritmu je nazývána jako *face ambiguity* (někdy také *external ambiguity*). Je způsobena tím, že pro stejnou konfiguraci vrcholů buňky lze použít více kombinací triangulací. Řešení tohoto problému bylo navrženo několik, od poměrně složitých postupů, které analýzou okolí rozhodují, která kombinace triangulace je vhodná [6], až po poměrně jednoduché řešení zakládající se na rozšíření triangulační tabulky[2]. Bylo zjištěno, že tyto nejednoznačnosti jsou způsobeny použitím reflexivní symetrie při výběru správné triangulace izoplochy. Pokud je reflexivita vynechána a využije se pouze rotační symetrie stoupne počet konfigurací na 23, viz obrázek 4.7. Toto by podle G. M. Nielsena [7] mělo být dostačující prevencí vzniku výše popsaných jevů. Druhým typem vad v triangulaci, které naštěstí nezpůsobují žádné nekonzistentnosti v topologii polygonálního modelu, jsou takzvané *internal ambiguity*. Jejich řešením může být opět rozšíření triangulační tabulky a rozsáhlejší analýza stavu okolí použitá pro volbu správné konfigurace, viz [3]. Nicméně, vzhledem k tomu, že je v předchozím kroku zpracování použit dilatační operátor a z po-

[1] V závislosti na tom, zda byl dodržen Shannonův-Nyquistův-Kotělnikovův vzorkovací teorém: $f_{vz} \geq 2 \cdot f_{max}$

[2] Poměrně rozsáhlý přehled článků týkajících se oprav a modifikací algoritmu Marching Cubes, za účelem zvýšení efektivity a docílení topologicky správné triangulace lze nalézt v souhrnném článku A survey of the marching cubes algorithm [5]



Obrázek 4.6: Standardní triangulační tabulka

pisu jeho chování je jasné, že může dojít ke sjednocení oddělených objektů, není řešení internal ambiguity nezbytně nutné, protože i tak není tento algoritmus jako celek schopen garantovat správnou topologii. Proto byla pro implementaci využita varianta G. M. Nielsena [7]. Následně, po výběru triangulace dojde k interpolaci hodnot jednotlivých vrcholů buňky tak, aby se vrcholy trojúhelníků, ležící na hranách buňky, nacházely přesně v místě s požadovanou izohodnotou. K nalezení souřadnic je využita lineární interpolace

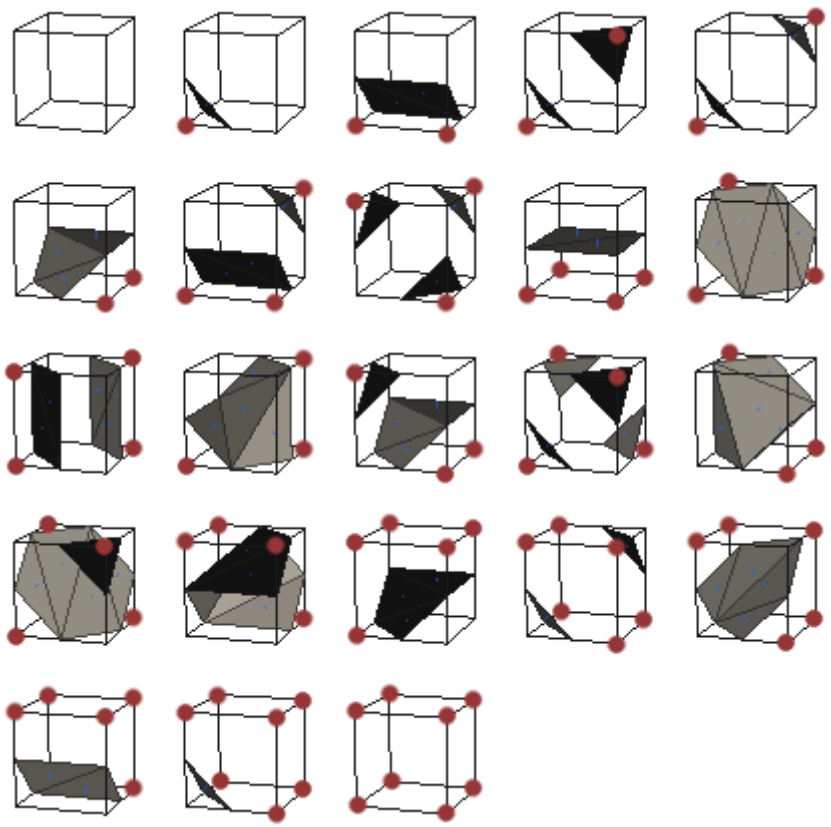
$$\mathbf{Q} = \mathbf{A} + (\mathbf{B} - \mathbf{A}) \cdot q \tag{4.3}$$

kde $q = \frac{h(\mathbf{Q}) - h(\mathbf{A})}{h(\mathbf{B}) - h(\mathbf{A})}$, kde $h(\mathbf{A}), h(\mathbf{B})$ a $h(\mathbf{Q})$ jsou hodnoty vzorků, resp. požadovaná izohodnota. Z chování algoritmu je zřejmé, že i pro rovné plochy vzniká velké množství trojúhelníků, nedochází k jakémukoli přizpůsobování křivosti extrahované izoplochy a celkový počet trojúhelníků bývá enormní. Přes všechny tyto nevýhody je tento algoritmus robustní a jednoduchý k implementaci (zvláště v případech, kdy není nutno tvořit transformační tabulky).

4.4 Decimace řízená délkou hrany

Dalším krokem algoritmu Hierarchical Iso-Surface Extraction je decimace polygonálního modelu, jež je výstupem Marching Cubes. Požadavkem na decimaci je především eliminace malých trojúhelníků, které nesou pouze mizivou geometrickou informaci, vznikajících v případě, že je izohodnota blízká hodnotám vzorků ve vrcholech buňky. Takové trojúhelníky vedou k degradaci kvality výsledného meshe a proto je výhodné je na začátku procesu zpracování polygonálního modelu odstranit. Snížení celkového počtu primitiv je až sekundárním efektem. Proto je zde použita velice jednoduchá decimace řízená délkou hrany trojúhelníku, pracující ve dvou krocích:

1. Každý trojúhelník, jehož hrana je kratší než $\alpha \cdot 2^l h$, kde α je uživatelsky definovaný koeficient a zároveň platí, že $\alpha > 0$, je smrštěn do svého těžiště, viz obrázek 4.8 vlevo.



Obrázek 4.7: Rozšířená triangulační tabulka

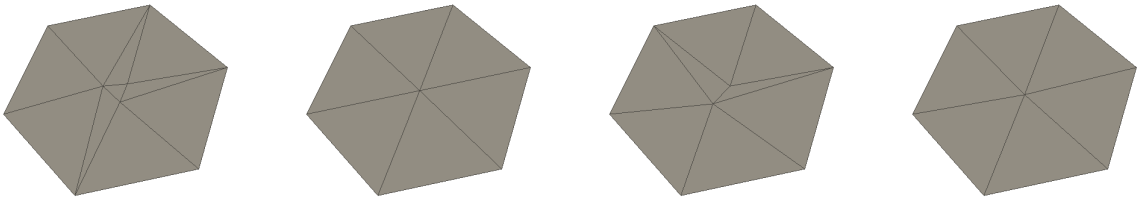
2. Následně jsou do svého středu smrštěny všechny zbývající hrany kratší než výše uvedený práh (na obrázku 4.8 vpravo).

V těchto případech může vyvstat situace, kdy takovýto kolaps hrany vede k vytvoření nonmanifold modelu, protože vznikají incidentní trojúhelníky, jež mají všechny tři vrcholy stejné, liší se pouze směrem normály, a tudíž vytvářejí nekonečně tenkou plochu. Všechny tyto trojúhelníky musí být odstraněny viz 4.9.

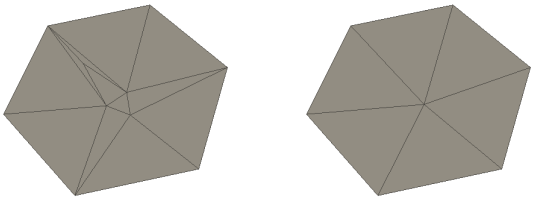
Vhodným nastavením parametru α je hodnota blízká 0,5, která vede ke snížení počtu trojúhelníků cca o 20 %.

4.5 Projekce izoplochy

Zdecimovaný polygonální model z předchozího kroku vlivem podvzorkování (zejména při použití dilatačního operátoru, viz obrázek 4.4) neleží na původní izoploše definované volumetrickými daty na úrovni f^0. Proto je nutné takový model na původní izoplochu promítnout. Aby byl algoritmus dostatečně robustní a nedocházelo k průnikům povrchu po projekci, je zde opětovně využita hierarchie objemových těles k postupnému promítání. Povrch je nejprve promítnut z úrovně l na úroveň $l-1$, dále na $l-2$, ..., až na úroveň $l=0$. Toto zaručuje, že bude povrch na úrovni $l-1$ vzdálen maximálně o délku $2^l h$, což odpovídá délce hrany dvou voxelů na úrovni $l-1$. Nalezení takové izoplochy však není triviální záleži-



Obrázek 4.8: Decimace trojúhelníku, decimace hrany



Obrázek 4.9: Decimace okolí trojúhelníku

tostí. Pro každý vrchol polygonálního modelu je vyslán paprsek v požadovaném směru, pro který je postupně vypočítán vždy bod průniku do a z buňky tvořené 8 voxely (tyto body jsou dále značeny jako A_{in} a A_{out}). V případě, že zpracovaný vrchol leží uvnitř objemového modelu, pak přirozeně také náleží určité buňce a pro tuto buňku nalezneme pochopitelně pouze jeden průnik v místě opuštění paprsku. Za těchto okolností je za A_{in} považován bod na souřadnicích zpracovávaného vrcholu. Naopak v případě, že vrchol v objemu neleží, což se může, při použití dilatačního filtru, stát, je nejprve potřeba určit bod průniku paprsku do obalového tělesa celého objemu a následně určit první buňku, kterou paprsek protíná. Vzhledem k tomu, že je celý volumetrický model zarovnán rovnoběžně se souřadným systémem, lze tyto průniky s výhodou vypočítat obdobně jako v případě výpočtu průniku paprsku s *Axis Aligned Bounding Boxem*. Pro každou ze tří bližších rovin krychle, které lze snadno určit ze směru jejich normály, je analyticky vypočítán bod průniku s paprskem

$$t = -\frac{\mathbf{n} \cdot \mathbf{org} + d}{\mathbf{n} \cdot \mathbf{dir}}$$

kde **org** je počáteční bod paprsku, **dir** je normalizovaný směrový vektor paprsku, **n** je normála plochy, a d je koeficient obecné rovnice roviny v prostoru $ax + by + cz + d = 0$. Bod průniku je pak $\mathbf{p} = \mathbf{org} + \mathbf{dir} \cdot t$. Ten bod, který je nejvzdálenější od počátku paprsku (tedy pro t_{max}) je posléze podroben testu, zdali leží v intervalech souřadnic definujících stěnu krychle. Obdobně lze postupovat při výpočtu A_{out}, s tím rozdílem, že je počítán průnik se třemi vzdálenějšími stěnami (s normálou směřující do téhož poloprostoru) a k testu je naopak vybrán průsečík nejbližší. Pro tyto dva body A_{in} a A_{out} jsou trilineární interpolací vypočítány jejich hodnoty vzorkované veličiny, které jsou znovu interpolovány vůči izohodnotě hledané izoplochy. Pokud bod Q s příslušnou izohodnotou leží na úsečce definované

body A_{in} a A_{out}, byl nalezen průnik s izoplochou na paprskem dané polopřímce právě v této buňce a je tak možno zpracovávaný vrchol v posunout do tohoto místa. V opačném případě, kdy hledaný bod Q leží mimo úsečku A_{in} a A_{out}, je potřeba pokračovat následující buňkou. Tento postup promítání je principiálně velmi podobný Raycastingu používanému jako DVR. Směr projekce vrcholu na isosurface, a tedy potažmo směr paprsku, jež je využit k výpočtu průsečíku, lze určit několika způsoby. První z nich je využití gradientu vzorkované hodnoty, který byl ale autory algoritmu zavržen, kvůli přítomnosti nežádoucího šumu. Proto jsou vrcholy promítnuty ve směru jejich normály, kterou lze získat jako střední hodnotu normál trojúhelníku v okolí prvního řádu. To zda bude průsečík vyhledán ve směru normály nebo proti směru je řešeno promítnutím na nejbližší nalezený průsečík.

4.6 Relaxace

Po promítnutí, může, přestože je prováděno postupně z nižší úrovně detailu objemového modelu na vyšší, dojít k shlukování vrcholů nebo dokonce k průnikům izoplochy. Proto za promítáním následuje krok vyhlazení modelu relaxačním operátorem. Nejjednodušším relaxačním operátorem je diskrétní aproximace *Laplaceova operátoru*

$$L(\mathbf{v}) = \frac{1}{|\text{adj}(\mathbf{v})|} \sum_{\mathbf{w} \in \text{adj}(\mathbf{v})} (\mathbf{w} - \mathbf{v}) \tag{4.4}$$

Nevýhodou tohoto operátoru je, že posouvá vrcholy i ve směru jejich normály (což není vhodná vlastnost, vzhledem k tomu, že vrcholy byly právě promítnuty a proto leží na požadované izoploše), a to má za následek smršťování modelu a především zaoblování hran. Protože je pouze potřeba redistribuovat vrcholy v rámci plochy na kterou byly promítnuty, je zde aplikován operátor využívající pouze tečnou část posunu daného Laplaceovým operátorem. Je definován vztahem

$$T(\mathbf{v}) = L(\mathbf{v}) - (L(\mathbf{v}) \cdot \mathbf{n_v}) \cdot \mathbf{n_v} \tag{4.5}$$

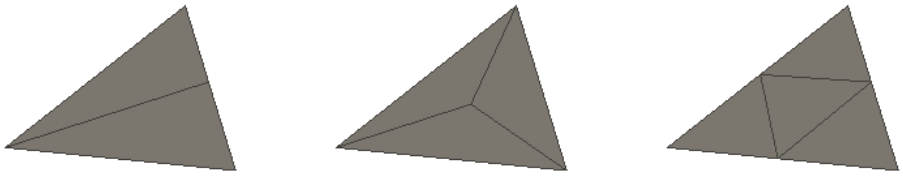
kde $\mathbf{n_v}$ je normála vrcholu $\mathbf{v}$.

4.7 Red-Green triangulace

Posledním logickým krokem procesu extrakce izoplochy tohoto algoritmu je zjemnění polygonální sítě v místech s vysokou křivostí pro zachycení patřičných detailů. Dělení trojúhelníků lze kategorizovat do tříd podle dvou kritérií. Jednak, zdali jde o adaptivní nebo plošné zjemňování, a jednak podle typu dělení rozlišujeme *1-to-2*, *1-to-3* a *1-to-4* dělení, viz obrázek 4.10.

V tomto algoritmu je pochopitelně požadováno adaptivní dělení typu 1-to-4, které zachovává dobrý poměr stran nově vzniklých trojúhelníků. Konkrétní použitý postup se nazývá *Red-Green* nebo také *Butterfly* triangulation a klade navíc omezení na rozdíly velikostí sousedních trojúhelníků, což má za následek postupné změny ve velikostech. Takový polygonální model se nazývá *balanced mesh*.

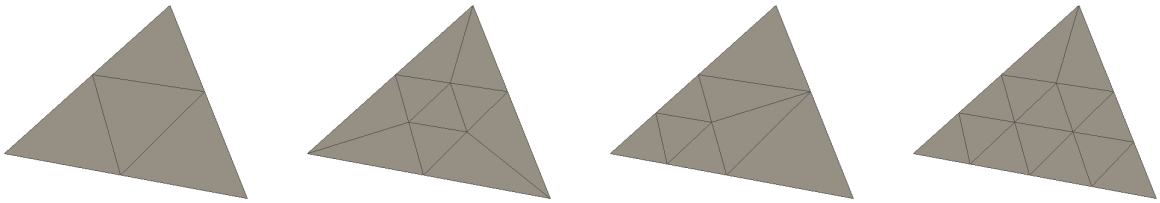
Jsou definovány dva typy trojúhelníku, zelené se sudým stupněm úrovně a červené s lichým. Pak při rozdělení zeleného trojúhelníku vznikají 4 nové zelené trojúhelníky, jejichž úroveň je o dva vyšší, než úroveň původního trojúhelníka. Zároveň vznikají 3 červené trojúhelníky (s o 1 vyšším stupněm úrovně), aby se předešlo *T-spojům* vzniklým na hranách



Obrázek 4.10: Dělení, zleva *1-to-2*, *1-to-3* a *1-to-4*

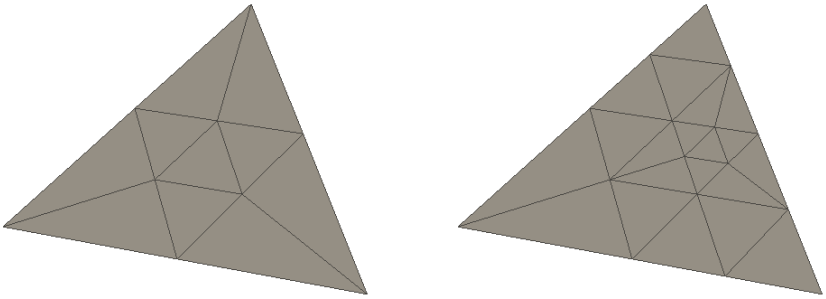
původního trojúhelníku (na obrázku 4.11 vlevo). Naopak při snaze opětovně rozdělit červený trojúhelník přicházejí v úvahu dvě možnosti:

1. Požadavek na rozdělení je na jiné, než na již rozdělené straně. Pak se trojúhelník stává zeleným na o 1 vyšší úrovni dělení – 4.11 vpravo.
2. Požadavek na rozdělení je na téže straně, která již byla rozdělena. V takovém případě se trojúhelník nejprve stává zeleným a v druhém kroku je jeden z nově vzniklých trojúhelníků opětovně rozdělen na dva červené (viz obrázek 4.12).



Obrázek 4.11: Levá dvojice: Green → Green, pravá dvojice: Red → Green

Zároveň platí, že se úrovně nesmí lišit o více něž o jeden stupeň, což klade další požadavky na kontrolu okolí. Nevýhodou tohoto postupu je, že dělení trojúhelníka není operace, kterou by nezasáhl do svého okolí právě kvůli vzniku T-spojů na hranách (na rozdíl od např. 1-to-3 dělení, kde když dojde k výpočtu rozdělení, každý trojúhelník je schopen se rozdělit aniž by ovlivnil své okolí). Toto lze v zásadě řešit dvěma způsoby. Buď lze vyrobit při dělení mezi sousedícími trojúhelníky *díry* a ty následně zacelit s tím, že pokud má trojúhelník, jenž se dělit neměl u své hrany jednu díru, pak se stává červeným, pokud jich je více než jedna, pak se rozdělí na zelený. Druhou možností je použít přístupu S. Seegera [10], který poukazuje na to, že každé dělení je možno realizovat pomocí operace *edge split* (1-to-2 subdivision) a *edge flip*, což provede přímé dělení okolních trojúhelníků na červené. Navíc, v případě, že je požadováno červený trojúhelník rozdělit z jiné strany, než ze které je aktuálně rozdělen, je toto možné udělat bez omezení a pouze dokončit jeho dělení stejným způsobem i na zbývající třetí hraně. Takto není potřeba vytvářet a zacelovat díry,



Obrázek 4.12: Red → Green → Red

které by případně vznikaly u T-spojů. Jediným problémem je dělení Red → Green → Red, které je nutno detekovat s předstihem a provést korekce před požadovaným dělením. To je v této implementaci řešeno pomocí algoritmu pro prohledávání stavového prostoru – tzv. *Breadth-first search*[3]. Tím je zaručeno že dojde k detekci problematických míst, kde by došlo k nesprávnému dělení a nebo k porušení omezení na úrovně incidentních trojúhelníků a v případě, že je takové místo nalezeno, je tento trojúhelník přidán na zásobník, což zaručí správnou posloupnost dělení jednotlivých trojúhelníků. Navíc to umožňuje vyhnout se nežádoucí rekurzi, která by v případě větší modelů obsahujících řádově 10^6 a více trojúhelníků mohla v určitých konfiguracích vést k přetečení zásobníku.

Toto dělení je použito na trojúhelníky, které jsou od hledané izoplochy v některém místě své plochy vzdálenější než zadané ϵ. Proto je potřeba každý trojúhelník T s vrcholy $\mathrm{u}, \mathrm{v}, \mathrm{w}$ navzorkovat v barycentrických souřadnicích $\alpha\mathrm{u} + \beta\mathrm{v} + \gamma\mathrm{w}$, kde $\alpha + \beta + \gamma = 1$. V této práci jsou použity celkem 4 vzorky, 3 na hranách trojúhelníka a jeden v jeho těžišti. Pro každý takový bod je potřeba opět najít průnik s hledanou izoplochou, v tomto případě je hledán ve směru normály daného trojúhelníka. Pokud je alespoň jeden vzorek vzdálen od izoplochy více než zadané ϵ, je tento trojúhelník rozdělen, a všechny nově (i nepřímo) vzniklé body jsou následně promítnuty na izoplochu a je provedeno jejich vyhlazení relaxačním operátorem. Toto zaručuje, že výsledek bude přesný balanced mesh s malými trojúhelníky v místech s velkou křivostí povrchu a velkými primitivy na rovných plochách. Celý postup od promítání, přes vyhlazování a zjemňování je iterativně opakován, dokud není polygonální model přesně promítnut na izoploše objemového modelu f^0.

[3] Česky prohledávání do šířky

Kapitola 5

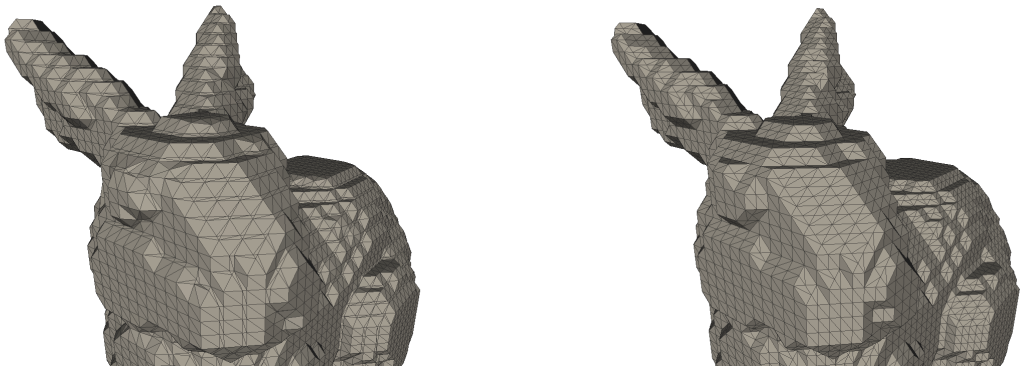
Návrhy modifikací algoritmu

Na základě testů a analýzy chování algoritmu byly navrženy dvě změny, které by mohly přispět ke zkvalitnění výsledků převodu a urychlení výpočtu.

5.1 Odstranění decimace a interpolace

Jediným argumentem pro zapojení decimace do celého procesu je odstranění trojúhelníků s nepodstatnou plochou. Tato primitiva způsobují špatný výchozí stav polygonálního modelu pro následné zjemňování pomocí Red Green triangulace, protože všechny trojúhelníky jsou na počátku označeny za zelené s úrovní 0, bez ohledu na jejich skutečnou velikost. Navíc tyto trojúhelníky nenesou patřičnou informaci o extrahovaném izopovrchu. To, že dojde ke snížení počátečního počtu trojúhelníkových primitiv lze považovat za pouhý vedlejší efekt, protože primárním nástrojem, který slouží k zajištění nízkého počtu primitiv v počátečním polygonálním modelu je podvzorkování v objemové oblasti. Navíc, jak bylo již dříve naznačeno, tyto buďto degradované, nebo celkově malé trojúhelníky vznikají především proto, že v rámci algoritmu Marching Cubes dochází k interpolaci vrcholů na hranách buněk, a to z toho důvodu, aby polygonální model co nejlépe odpovídal extrahované izoploše. Tato potřeba ale v případě tohoto postupu nevzniká, protože vrcholy na počátku stejně neleží na izoploše nacházející se v obejmu f^0 jsou postupně několikrát promítány na požadované pozice.

Proto je první navržená změna oproti původnímu algoritmu prostá. V případě, že by algoritmus Marching Cubes neinterpoloval hodnoty vrcholů trojúhelníků, výsledná primitiva nebudou degradovaná (detail na obrázku 5.1) a tudíž nevzniká potřeba zavádět decimaci do celého procesu. Tím se jednak ušetří výpočetní čas interpolace v MC, a jednak výpočet samotné decimace, kterou navíc nebude potřeba implementovat. Nehledě na to, že pokud je izohodnota blízká středu intervalu mezi hodnotami vně a uvnitř extrahovaného tělesa, pak i při využití interpolace leží vrcholy primitiv na středech hran buněk. Trojúhelníky jsou tak dobře tvarované a proto jich decimace zlikviduje jen mizivé procento. Proto byl algoritmus MC implementován i s možností vypnutí interpolace, kdy jsou vrcholy umístěny přesně do středu hran buněk a v celém procesu HIE lze vypnout decimaci, aby bylo tuto hypotézu možno ověřit v rámci testování.

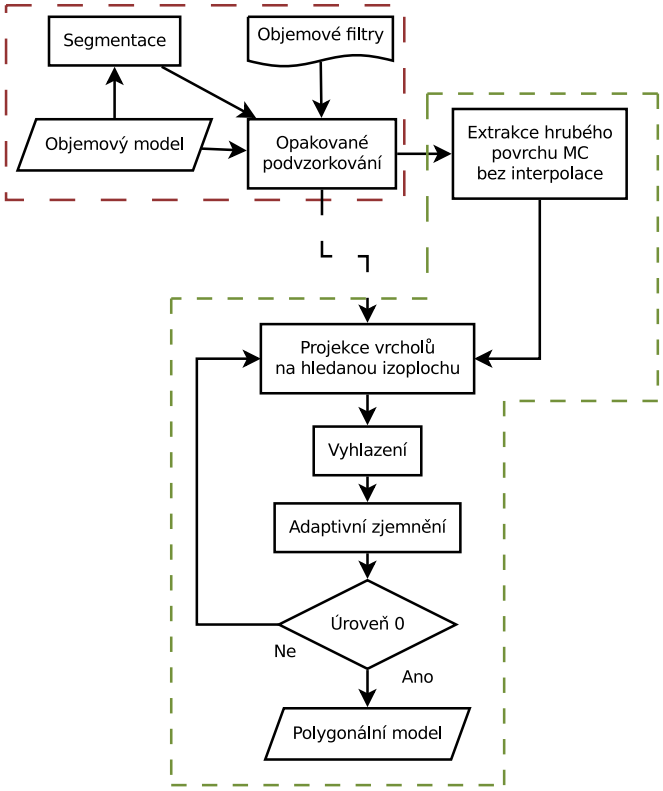


Obrázek 5.1: Marching Cubes: Vlevo s použitím interpolace, vpravo bez interpolace

5.2 Zavedení jednoduché segmentace ve spojení s dilatačním operátorem

Jak bylo již nastíněno dříve i dilatační operátor má, zvláště při použití na reálných datech své nevýhody – zaplňuje drobné otvory a způsobuje splynutí dříve oddělených izoploch v objemovém modelu na vyšší úrovni hierarchie. Řešením tohoto nežádoucího chování by mohlo být zapojení jednoduché segmentace a úprava dilatačního operátoru. Segmentační proces by na základě prahování s izohodnotou pro každý voxel určil, zda tvoří vnitřní část extrahovaného modelu a nebo je okolním prostředím. Následně by se na takto naprahovaných datech provedla jednoduchá segmentace k určení počtu objektů a příslušnosti jednotlivých voxelů do těchto objektů. Dilatační operátor by se následně choval tak, že by v případě, že vrcholy buňky leží v různých třídách ekvivalence segmentace, vložil vynucenou podprahovou hodnotu pro zachování separace objektů. Nutno podotknout, že i toto řešení za určitých okolností selhává a nezamezí vymizení děr z tělesa. Nicméně mohlo by zlepšit stávající situaci a díky tomu, že je z pravidla potřebná jen nízká hierarchie volumetrických modelů, by mohlo toto řešení být dostačující. Výsledný proces extrakce je zobrazen na 5.2.

Z časových důvodů však toto řešení nebylo realizováno a tak jeho přínos nelze vyvrátit ani potvrdit testy.



Obrázek 5.2: Proces upraveného výpočtu Hierarchical Iso-Surface Extraction

Kapitola 6

Implementace Hierarchical Iso-Surface Extraction

V rámci této kapitoly bude ve zkratce popsán výběr implementačních nástrojů, knihoven a nastíněny některé implementační zajímavosti.

6.1 Výběr programovacího jazyka, podpůrných knihoven a nástrojů

Jako hlavní implementační jazyk celého projektu bylo, jako již tradičně, zvoleno *C++*, pro svou rozšířenost, a tedy i dostupnost dílčích knihoven pro práci s objemovými a polygonálními modely a také díky své rychlosti a efektivnosti, která vyplývá z faktu, že se jedná o kompilovaný jazyk. Další důležitou roli hrálo samozřejmě to, že jde o objektově orientovaný jazyk umožňující kvalitní, zapouzdřitelný a lehce udržovatelný návrh celé aplikace. V neposlední řadě bylo využito možnosti generického programování pomocí *template*, které díky časné vazbě[1] nezatěžují výpočetní čas zbytečnou režií spojenou s vazbou pozdní.

V rámci projektu bylo využito dvou relativně velkých, stále udržovaných a moderně navržených knihoven. První z nich je knihovna pro práci s objemovými modely, původně zaměřená na segmentaci medicínských dat, Medical Data Segmentation Toolkit (MDSTk) ve verzi 1.1.1, která je využita taky jako základní stavební prvek pro tvorbu jednotlivých programových modulů (dostupná online z URL: `http://mdstk.sourceforge.net/`). Druhou knihovnou je OpenMesh (online na `http://www.openmesh.org/`, aktuální verze 2.3.1), využitá pro práci s polygonálními modely, založená na datové struktuře *halfedge* 2.2, umožňující rychle využívat topologické informace o generovaném povrchu. Tato knihovna je rovněž velice moderně navržena s využitím návrhových vzorů, je velmi efektivní a modulární díky široce rozšířenému použití template a je stále aktivně podporována. Nutno podotknout, že tyto knihovny byly využity pouze jako implementace datových struktur a manipulace s nimi (načítání a ukládání do souborů), s výjimkou MDSTk, odkud byl navíc také využit princip tvorby programových modulů. Veškeré další algoritmy spojené s implementací Hierarchical Iso-Surface Extraction byly implementovány autorem, přestože např. decimace řízená délkou hrany je podporována v knihovně OpenMesh.

Mezi další jazyky použité při vypracovávání tohoto projektu patří *python*, který byl využit zejména ve skriptu generujícím triangulační tabulky, v rámci testovacích skriptů

[1] Z anglického Early binding

a drobných nástrojů generujících kostry modulů. Dále *BASH* k tvorbě testovacích a spouštěcích skriptů (tyto skripty je možno samozřejmě pouštět i na operačních systémech MS Windows, přestože vývoj probíhal výhradně na Linuxu), které byly z počátku duplikovány *BAT* skripty, od čehož bylo ale následně upuštěno. Jako poslední v řadě byl využit *JavaScript*, k přípravě drobného grafického nástroje umožňujícího vizuálně zavést triangulační případy pro Marching Cubes a objemový prohlížeč.

Vývojovým prostředím bylo *NetBeans* s plnou podporou C++, pro zpracování výsledků je použit *gnuplot*, *meshlab* a *blender*. Programovou dokumentaci je možno vygenerovat pomocí nástroje *Doxygen*.

6.2 Implementace

Aplikace je navržena jako soubor modulů bez GUI. Hlavním modulem je `grExtractor` umožňující extrakci izoploch z objemových dat, buďto s použitím Marching Cubes, nebo Hierarchical Iso-Surface Extraction, pro různá nastavení. Druhý modul `grStats` pak analyzuje výstupní polygonální model z hlediska jeho kvality, podrobnější informace o vyhodnocovaných vlastnostech se nacházejí v kapitole 7. Dále je vytvořena sada menších modulů, které slouží jako *testbench*, případně jako jednotkové testy pro jednotlivé dílčí části algoritmu, které vzhledem k jednoduché dekompozici byly implementovány nezávisle na sobě. Tyto testovací moduly zároveň posloužily ke generování ilustrací v této práci.

Ke spouštění jednotlivých testů byl vždy připraven ad-hoc skript, v závislosti na tom, která vlastnost testovaného algoritmu byla evaluována. Pro zpracování výstupních statistických dat z modulu `grStat` jsou ve složce test dva skripty. Skript `Histogram.sh` vytváří s pomocí gnuplot buď histogramy, to pokud je zadán pouze jeden statistický soubor ke zpracování a nebo lineárně aproximované histogramy, v případě, že je požadována společná analýza více datových souborů. Druhý skript `Table.py` pak slouží k vytvoření LaTeXových tabulek pro jejich snadnou prezentaci.

Samotný návrh tříd nebyl nijak složitý, vzhledem k tomu, že algoritmus je přímo popsán jako logická posloupnost po sobě jdoucích kroků, které byly tedy implementovány separátně (i s ohledem na to, aby byly případně použitelné v jiné aplikaci) a následně agregovány ve třídě HIE. Celý kód je, především kvůli své složitosti, řádně komentován. Programovou dokumentaci, včetně jednotlivých vazeb, lze vygenerovat nástrojem Doxygen.

Během fáze prototypování byl také navíc naprogramován modul pro převod polygonální reprezentace na objemový model, za účelem testování věrohodnosti a kvality extrakce objemu následným porovnáním s originálním polygonálním modelem. Základním algoritmem nutným pro implementaci takového modulu je výpočet průniku paprsku s trojúhelníkem v prostoru, který byl implementován podle [1]. Protože se ale jednalo pouze o prototyp, který navíc na rozdíl od celé aplikace používal *VectorEntity*[2] jako podporu pro polygonální modely, stejně jako ostatní moduly v raných fázích vývoje, není zdrojový kód tohoto modulu k dispozici. Jím vzorkované povrchy naopak jsou k nalezení v příloze.

[2]Knihovna pro podporu polygonálních modelů integrovaná do MDSTk, která však už není dále vyvíjena a podporována

Kapitola 7

Testování výsledků extrakce

Během testování bylo na výstupních polygonálních modelech měřeno několik veličin. Vedle očekávaného počtu primitiv (trojúhelníků, jejich hran a počtu vrcholů) je dále pro každý trojúhelník největší normovaný rozdíl ($\cdot 100$ v %) ploch s incidentními trojúhelníky. Toto měření má za úkol zdokumentovat, zdali dochází k postupným a nebo skokovým změnám ve velikostech trojúhelníků. Pochopitelně jsou z hlediska kvality polygonální sítě lepší změny postupné. Dále dochází k měření největší odchylky normál aktuálně analyzovaného trojúhelníka a jeho přilehlých sousedů (ve $^\circ$ v rozsahu $\langle 0; 180 \rangle$). Na této veličině lze pozorovat, zda jsou křivé plochy triangulovány s drobnými změnami ve sklonu trojúhelníků anebo zda dochází ke zlomové triangulaci. Zde je samozřejmě zapotřebí důkladná analýza a znalost extrahovaného objemu, protože se přirozeně v extrahovaném objektu mohou nacházet také ostré hrany. Poslední měřenou veličinou je velikost nejostřejšího vnitřního úhlu (opět ve $^\circ$, v intervalu $\langle 0; 45 \rangle$). Toto kritérium prozrazuje, zdali mají primitiva spíše tvar blízký rovnostrannému trojúhelníku, nebo zda jsou protáhlá a degradována. Za degradovaný trojúhelník se považuje každý s alespoň jedním vnitřním úhlem menším než 5 $^\circ$. Z každé z těchto veličin je vypočítána střední hodnota, směrodatná odchylka a extrémní hodnota pro celý polygonální model. V případě velikosti vnitřních úhlů je pak archivován počet degradovaných trojúhelníků.

7.1 Testování vlivu hloubky hierarchie na výsledný model

Na několika objemech byl testován vliv velikosti hierarchie na výslednou triangulaci modelu. Z výsledků je patrné, že příliš nízká hierarchie vede k velkému množství primitiv v extrahovaném tělese, protože počet voxelů není dostatečně redukován. Je možno konstatovat, že v případě velikosti hierarchie $0, 1$ a 2 výsledný model počtem trojúhelníků a jejich distribucí blízký výsledků algoritmu MC. Výsledná kvalita je pochopitelně lepší vzhledem k promítací a vyhlazovací fázi v algoritmu Hierarchical IsoSurface Extraction, ale z pohledu množství primitiv je model předimenzovaný a to do té míry, že nově generovaných trojúhelníků ve fázi adaptivního zjemňování je naprosté minimum. Trojúhelníky jsou navíc distribuovány téměř rovnoměrně, bez ohledu na křivost povrchu. Jako ideální se ukazuje hloubka hierarchie 3 až 5, kdy už je podvzorkování dostačující k patřičné redukci počtu trojúhelníků generovaných MC a zároveň není tento počet příliš malý, což vede ke nadbytečnému užití adaptivního dělení, které je přece jen pomalejší, než generování triangulace na základě tabulek. A v neposlední řadě je pak potřeba vícenásobné promítání. Vyšší úroveň hierarchie tak nepřináší žádné zlepšení z hlediska kvality výsledného polygonálního modelu, naopak

extrakce trvá déle (při úrovni 7 dokonce došlo k chybnému promítání, což má za následek shluk protínajících se trojúhelníků ve výsledném modelu – toho si lze povšimnout v tabulce 7.1 u extrému poměru velikostí). Prezentovaná data jsou pořízena na objemovém modelu králíka (rozměr objemu ve směru osy `x: 730`, `y: 566`, `z: 723`) získaného vzorkováním polygonálního modelu dostupného ve Stanfordském repositáři.

Název	**Počet primitiv**	**Rozdíl velikostí trojúhelníků**[1] ·100 [%]		**Odchylka normál**[1] ⟨0,0; 180,0⟩ [°]		**Velikost nejostřej. úhlu**[2] ⟨0,0; 45,0⟩ [°]	
	Vrcholy / **Trojúhel.**	**Průměr** σ[5]	**Extrém**[3]	**Průměr** σ[5]	**Extrém**[3]	**Průměr** σ[5]	**Degrad. tr.**[4]
Depth2	78604 157204	0,856678 1,466999	96,66661	8,439771 7,047095	175,30597	34,854614 3,064970	0
Depth3	22056 44108	0,878285 1,107672	30,10401	11,319223 9,507100	134,53984	34,792217 3,246228	0
Depth4	17336 34668	0,908545 1,164796	28,23538	13,127929 10,137598	160,51309	34,811778 3,616606	0
Depth5	17934 35864	0,818916 1,047331	30,37124	13,074977 10,326901	92,72721	34,638956 3,735802	0
Depth6	20158 40312	1,405032 14,421969	1782,6944	21,698035 32,432215	179,96015	35,597024 4,422884	0

Kvalita triangulace je dobrá, střední hodnota nejostřejšího vnitřního úhlu se pohybuje okolo 35 °. Lze si také povšimnout, že mezi úrovněmi 3–6 nedochází k další redukci počtu trojúhelníků ve výsledném meshi. Na grafu 7.1 pak lze vidět, že ke změnám úhlů dochází postupně, což je pro tento model bez ostrých hran v pořádku a dokazuje to správné promítání vrcholů bez přítomnosti šumu.

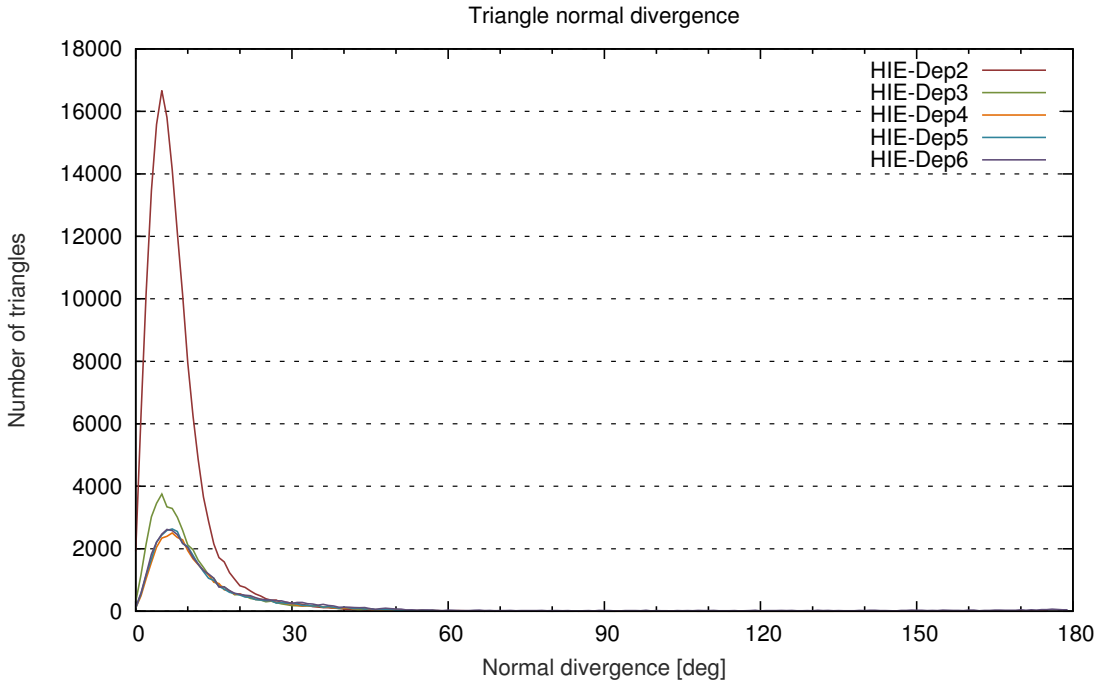
7.2 Testování vlivu hloubky hierarchie na výsledný model

Další test evaluoval vliv vynechání interpolace a decimace z celého procesu výpočtu triangulace. Na základě vizuální inspekce a analýzy výsledků lze o testovaných modelech prohlásit, že neinterpolované modely bez decimace jsou mírně větší co do počtu primitiv, což lze snadno suplovat zvýšením hierarchie, nicméně jejich kvalita je naprosto srovnatelná s modely, jež projdou interpolací hodnot v MC a následnou decimací, viz tabulku 7.2, graf změn velikostí trojúhelníků 7.2 a graf vnitřních úhlů 7.3, které dokazují, že se žádné degenerované trojúhelníky ve výsledném polygonálním modelu nenacházejí.

7.3 Porovnání s Marching Cubes

V rámci posledních testů bylo prováděno porovnání s algoritmem Marching Cubes. Typický fenomén tohoto algoritmu je jeho šum, a hůře tvarované trojúhelníky, které lze zřetelně

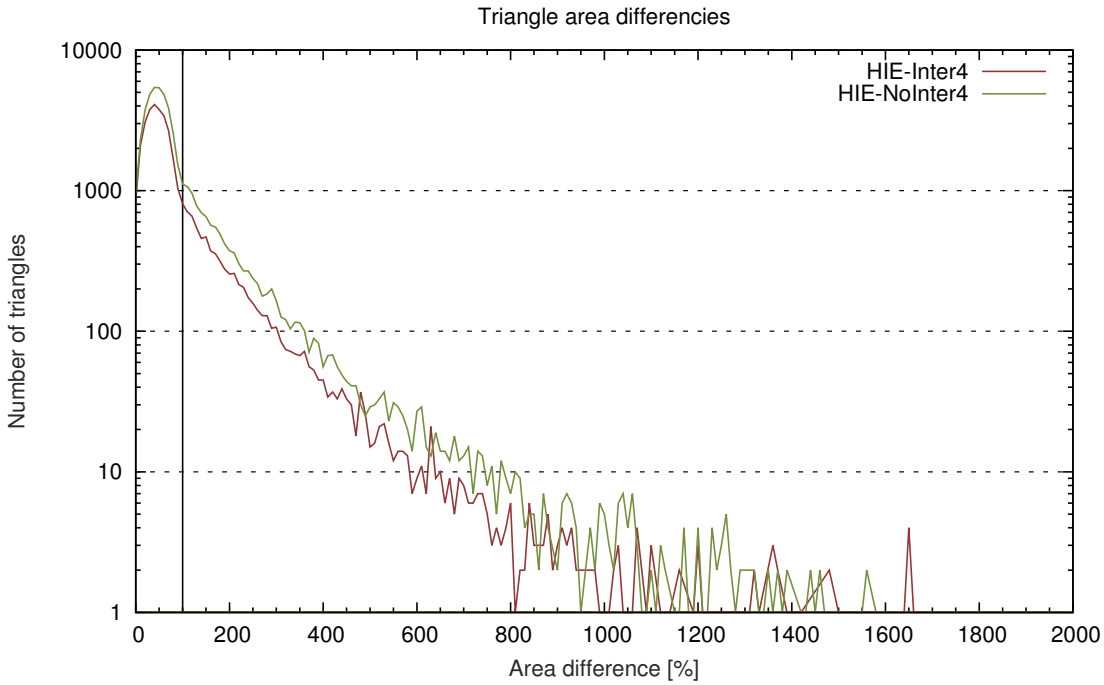
[1] Pro každý trojúhelník je brána v potaz maximální hodnota z hodnot vypočtených pro 3 incidentní trojúhelníky.
[2] Pro každý trojúhelník je vypočtena hodnota nejostřejšího úhlu.
[3] Globální extrém pro celý mesh
[4] Za degradovaný trojúhelník se považuje každý obsahující úhel menší než 5 °.
[5] Směrodatná odchylka (s_x)



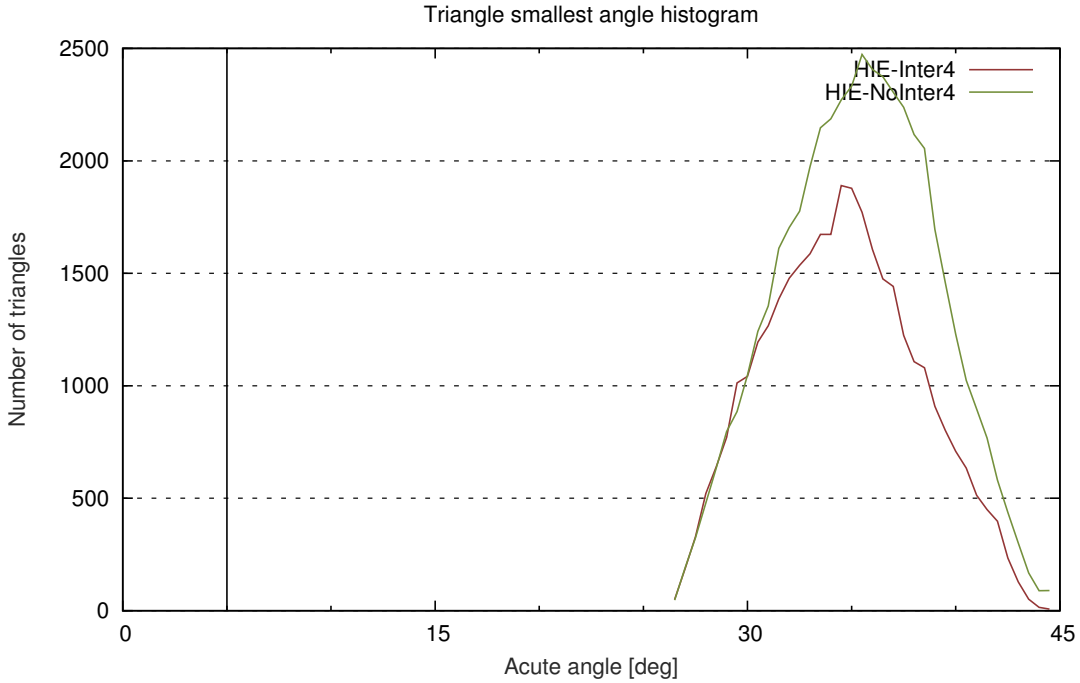
Obrázek 7.1: Test vlivu hloubky hierarchie

Název	**Počet primitiv**	**Rozdíl velikostí trojúhelníků**[1] ·100 [%]		**Odchylka normál**[1] ⟨0,0; 180,0⟩ [°]		**Velikost nejostřej. úhlu**[2] ⟨0,0; 45,0⟩ [°]	
	Vrcholy / **Trojúhel.**	**Průměr** σ[5]	**Extrém**[3]	**Průměr** σ[5]	**Extrém**[3]	**Průměr** σ[5]	**Degrad. tr.**[4]
Inter	17336 34668	0,908545 1,164796	28,23538	13,127929 10,137598	160,51309	34,811778 3,616606	0
NoInter	23856 47708	1,045275 9,427137	1978,0930	12,169310 13,526410	179,51313	35,583666 3,625011	0

vidět na histogramu vnitřních úhlů 7.4 v podobě maxima na 35 °. Vzhledem k značným rozměrům objemového modelu (`x: 894`, `y: 400`, `z: 631`) vizuálně daný šum příliš nevadí, velikost souboru a obtížná manipulace s ním, vzhledem k počtu primitiv, však ano. Naopak na histogramu 7.5 lze vidět, přirozenou změnu velikostí sousedních trojúhelníků (osa y je v logaritmickém měřítku). Vybrané obrazové výstupy jsou v příloze.

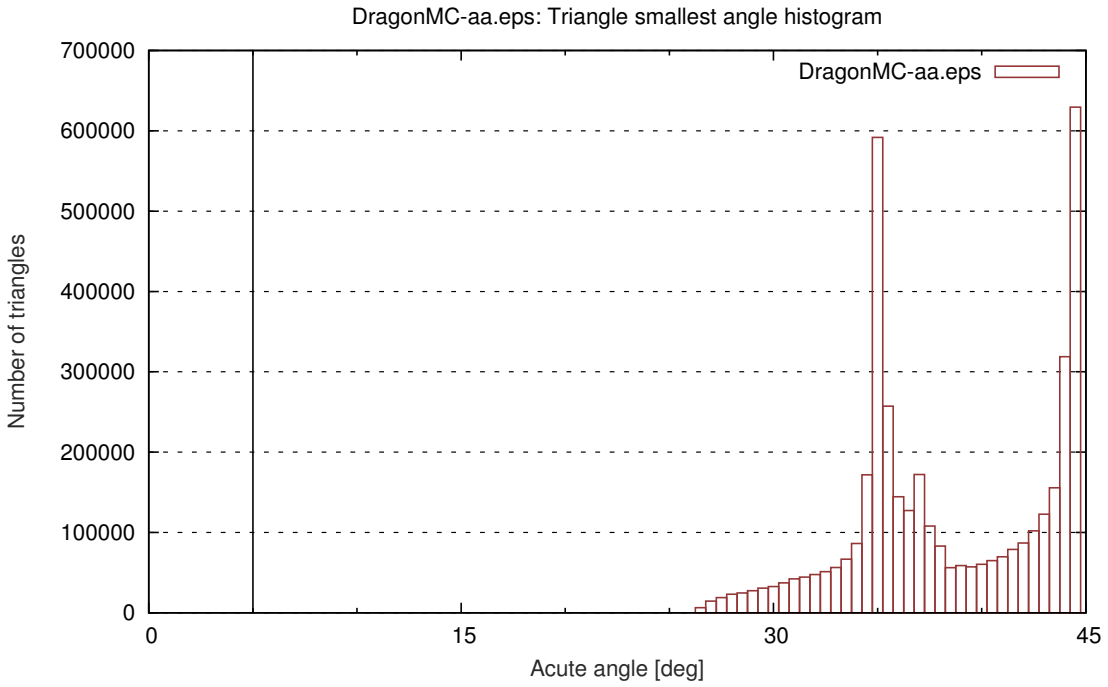


Obrázek 7.2: Test interpolace a decimace: změny plochy

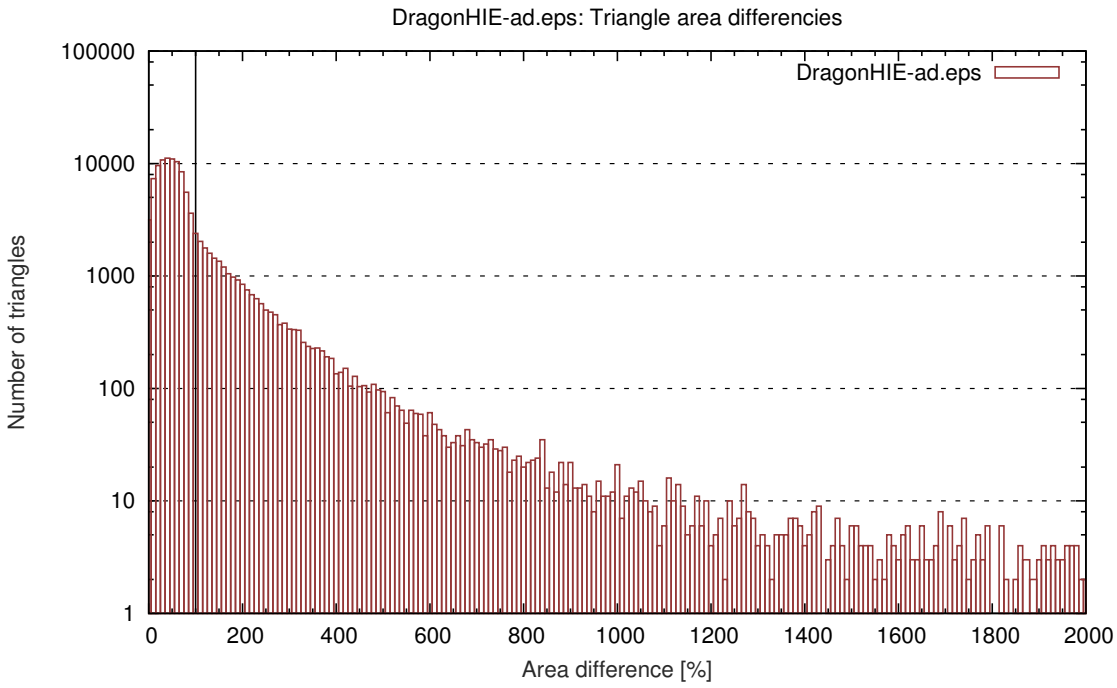


Obrázek 7.3: Test interpolace a decimace: ostré úhly

Název	Počet primitiv	Rozdíl velikostí trojúhelníků[1] ·100 [%]		Odchylka normál[1] ⟨0,0; 180,0⟩ [°]		Velikost nejostřej. úhlu[2] ⟨0,0; 45,0⟩ [°]	
	Vrcholy	Průměr	Extrém[3]	Průměr	Extrém[3]	Průměr	Degrad. tr.[4]
	Trojúhel.	σ[5]		σ[5]		σ[5]	
DragHIE	53836 107668	1,190760 6,732221	1036,0580	19,839314 24,296491	179,98555	35,423242 4,036420	0
DragMC	2064074 4128284	2130,856272 180802,63	57456387	16,408664 11,897922	155,34068	38,625395 4,700773	0



Obrázek 7.4: Marching Cubes: Ostré úhly



Obrázek 7.5: Hierarchical Iso-Surface Extraction: Změna velikostí

Kapitola 8

Závěr

Na základě naměřených výsledků se ukazuje, že největším hendikepem algoritmu Hierarchical Iso-Surface Extraction je využití dilatačního operátoru, což znemožňuje jeho praktické využití např. na medicínských datech, kde je přítomen šum a drobné separátní objekty, případně otvory, které tento operátor zaceluje. Řešení tohoto problému se ukázalo jako netriviální a tak lze říci, že je tento algoritmus vhodný spíše k extrakci plných nezašuměných objektů, kde ale dosahuje výborných výsledků ve srovnání s tradičním algoritmem Marching Cubes. Primitiva po extrakci mají tvar blízký rovnostranným trojúhelníkům, jsou adaptivně rozdistribuována na základě křivosti extrahovaného povrchu a tak je na rovných plochách pouze malé množství primitiv. Celkový počet trojúhelníků je oproti Marching Cubes takřka mizivý, což se kladně projeví na velikosti souborů a také na rychlosti manipulace s nimi.

Stanovených cílů, a sice seznámení se s problematikou a implementace netriviálního extrakčního algoritmu, se podařilo dosáhnout. V rámci této diplomové práce bylo nastudováno větší množství algoritmů pro extrakci izoploch z volumetrických dat a jedna z nich Hierarchical Iso-Surface Extraction byla plně implementována. Byla vybrána vhodná testovací kritéria a vlastnosti, a algoritmus byl podroben detailní evaluaci.

Rozšíření této práce je nasnadě. Extrakčních postupů existuje ještě celá řada a tak by bylo zajímavé doplnit další implementace a provést křížovou evaluaci vlastností algoritmů. Zároveň lze dále pracovat se současnou implementací, je zde určitý prostor pro optimalizace, řešení jeho nedostatků v podobě dilatace a případná paralelizace vhodných částí algoritmu.

Literatura

[1] Kensler, A. a Shirley, P. Optimizing Ray-Triangle Intersection via Automated Search. In *Interactive Ray Tracing 2006, IEEE Symposium on*. 2006. S. 33–38. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4061543>. ISBN 1-4244-0693-5.

[2] Labsik, U., Hormann, K., Meister, M. et al. Hierarchical Iso-Surface Extraction. *Journal of Computing and Information Science in Engineering* [online]. 2002, roč. 2, č. 4 [cit. 16. května 2013]. S. 323–329. Dostupné z URL: <http://link.aip.org/link/?CIS/2/323/1>.

[3] Lewiner, T., Lopes, H., Vieira, A. W. et al. Efficient implementation of Marching Cubes' cases with topological guarantees. *Journal of Graphics Tools*. 2003, roč. 8. Dostupné z URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.8509>.

[4] Lorensen, W. E. a Cline, H. E. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* [online]. New York, New York, USA: ACM, 1987 [cit. 16. května 2013]. S. 163–169. SIGGRAPH '87. Dostupné z URL: <http://doi.acm.org/10.1145/37401.37422>. ISBN 0-89791-227-6.

[5] Newman, T. S. a Yi, H. A survey of the marching cubes algorithm. *Computers & Graphics*. 2006, roč. 30, č. 5. S. 854–879. Dostupné z URL: <http://www.sciencedirect.com/science/article/pii/S0097849306001336>. ISSN 0097-8493.

[6] Nielson, G. M. a Hamann, B. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of the 2nd conference on Visualization '91* [online]. Los Alamitos, California, USA: IEEE Computer Society Press, 1991 [cit. 16. května 2012]. S. 83–91. VIS '91. Dostupné z URL: <http://dl.acm.org/citation.cfm?id=949607.949621>. ISBN 0-8186-2245-8.

[7] Nielson, G. M., Huang, A. a Sylvester, S. Approximating Normals for Marching Cubes applied to Locally Supported Isosurfaces. In *Visualization, 2002. VIS 2002. IEEE*. 2002. S. 459–466. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1183808>. ISBN 0-7803-7498-3.

[8] Poston, T., Wong, T.-T. a Heng, P.-A. Multiresolution Isosurface Extraction with Adaptive Skeleton Climbing. *Computer Graphics Forum* [online]. 1998, roč. 17, č. 3

[cit. 16. května 2013]. S. 137–147. Dostupné z URL: <http://dx.doi.org/10.1111/1467-8659.00261>. ISSN 1467-8659.

[9] Schreiner, J., Scheidegger, C. a Silva, C. High-Quality Extraction of Isosurfaces from Regular and Irregular Grids. *IEEE Transactions on Visualization and Computer Graphics* [online]. September 2006, roč. 12 [cit. 16. května 2012]. S. 1205–1212. Dostupné z URL: <http://dx.doi.org/10.1109/TVCG.2006.149>. ISSN 1077-2626.

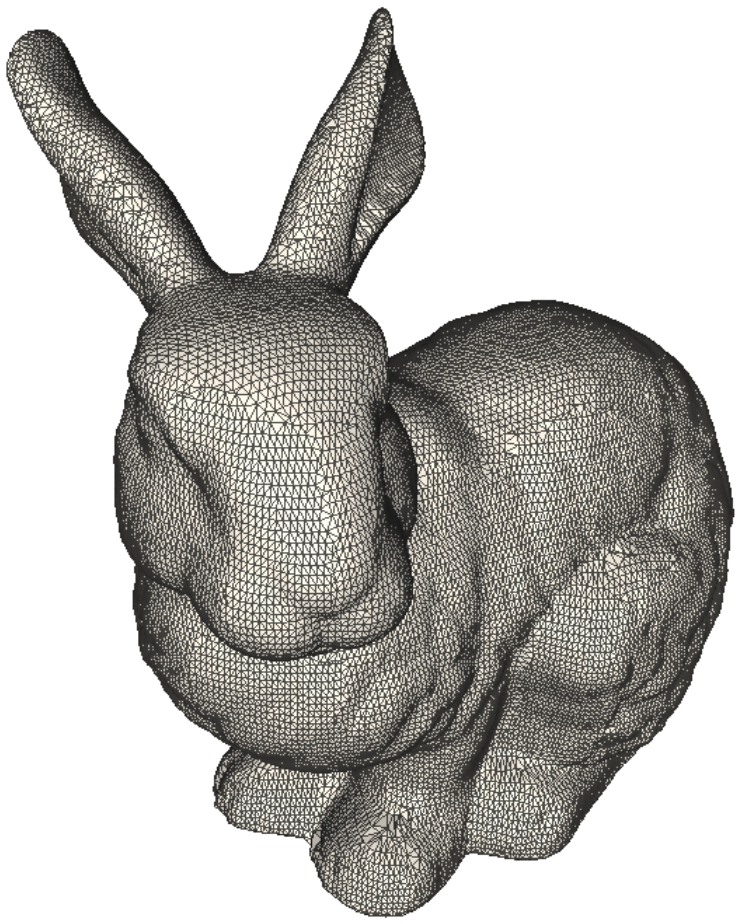
[10] Seeger, S., Hormann, K., Häusler, G. et al. A Sub-Atomic Subdivision Approach. In *Proceedings of the Vision Modeling and Visualization Conference 2001*. 2001. S. 77–86. VMV '01. Dostupné z URL: <http://dl.acm.org/citation.cfm?id=647260.718494>. ISBN 3-89838-028-9.

[11] Wood, Z. J., Schröder, P., Breen, D. et al. Semi-regular mesh extraction from volumes. In *Proceedings of the conference on Visualization '00*. Los Alamitos, California, USA: IEEE Computer Society Press, 2000. S. 275–282. VIS '00. Dostupné z URL: <http://dl.acm.org/citation.cfm?id=375213.375254>. ISBN 1-58113-309-X.

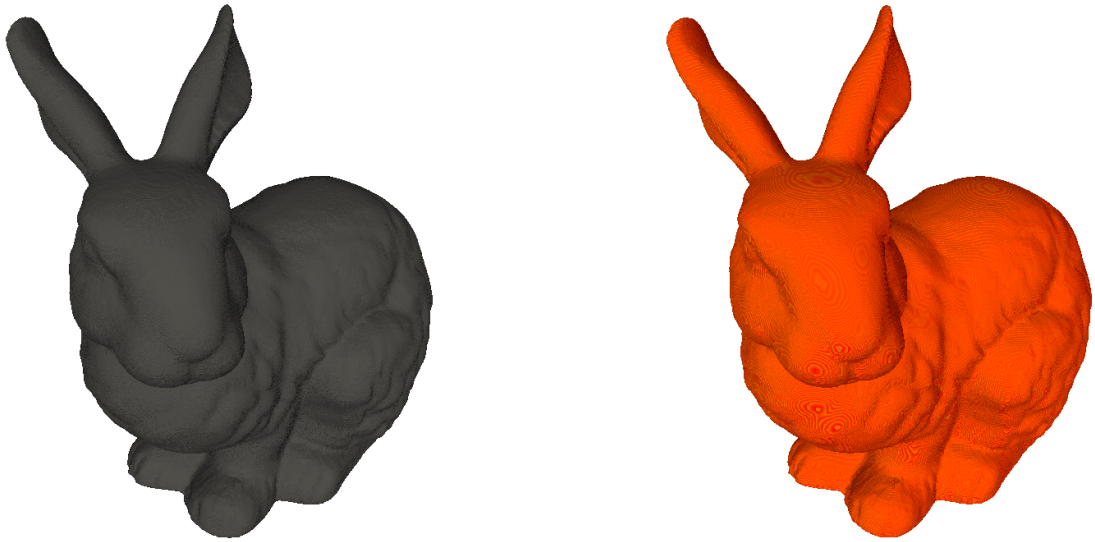
[12] Žára, Jiří and Beneš, Bedřich and Sochor, Jiří and Felkel, Petr. *Moderní počítačová grafika*. 2. Brno: Computer Press, 2004. 609 s. ISBN 80-251-0454-0.

Příloha A

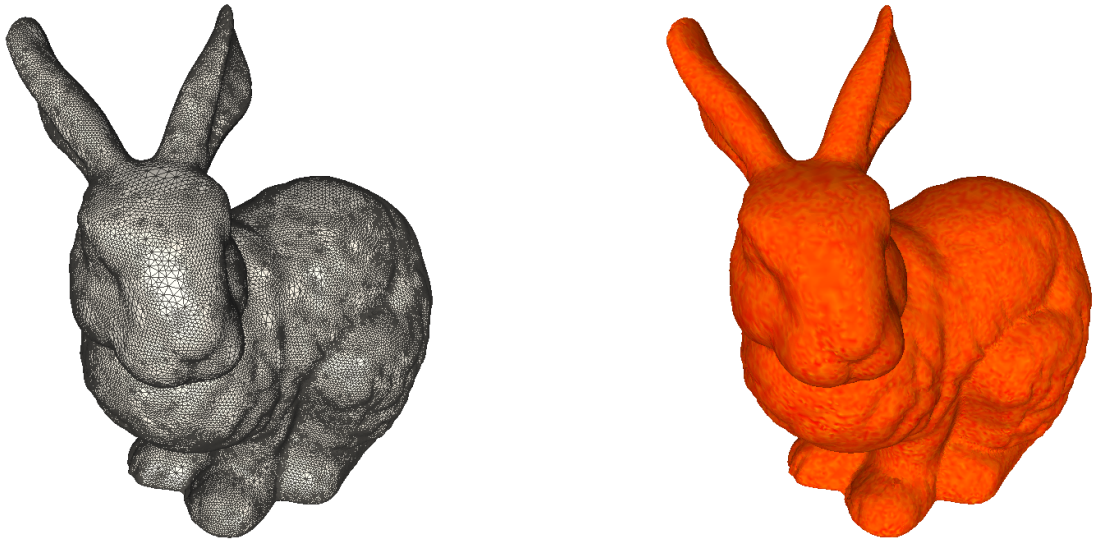
Obrazové výsledky vyhlazování



Obrázek A.1: Originální model

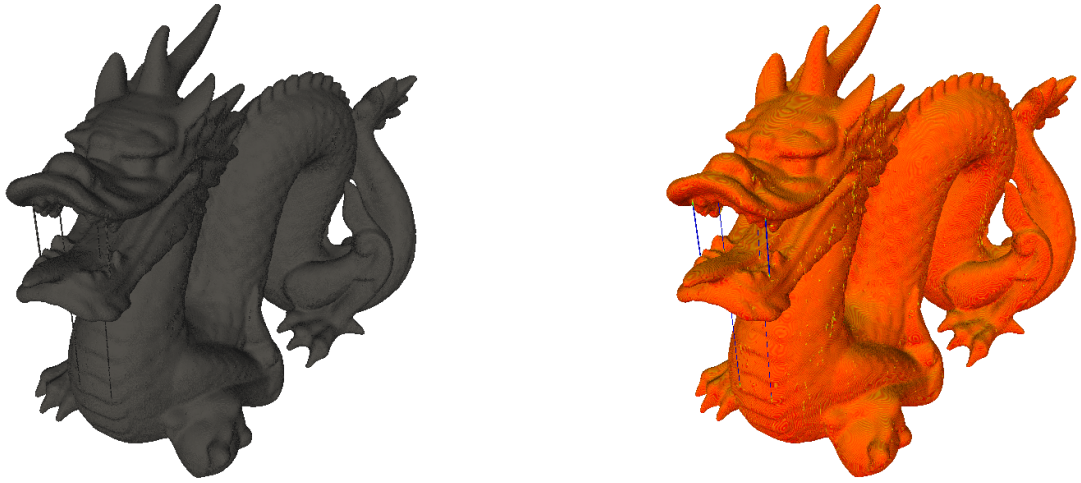


Obrázek A.2: Marching Cubes

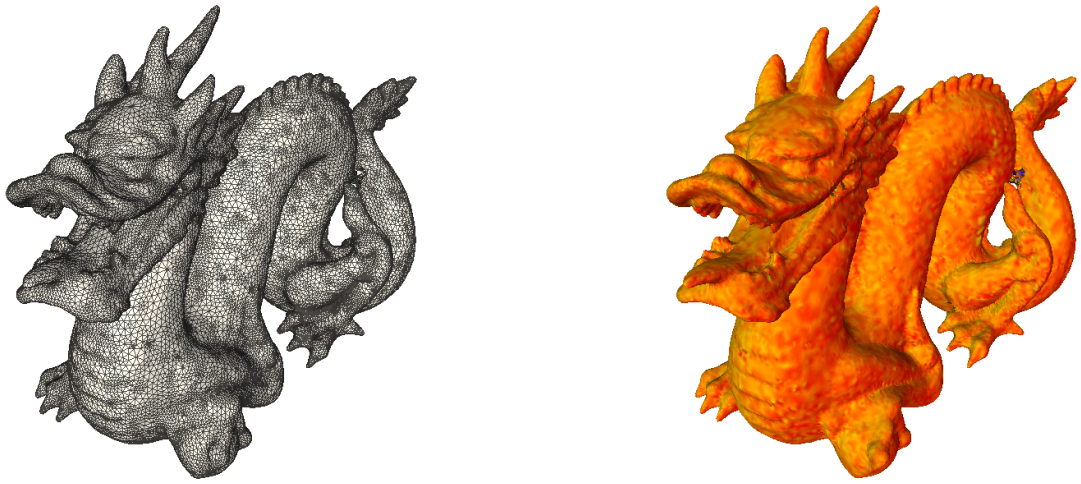


Obrázek A.3: Hierarchical Iso-Surface Extraction

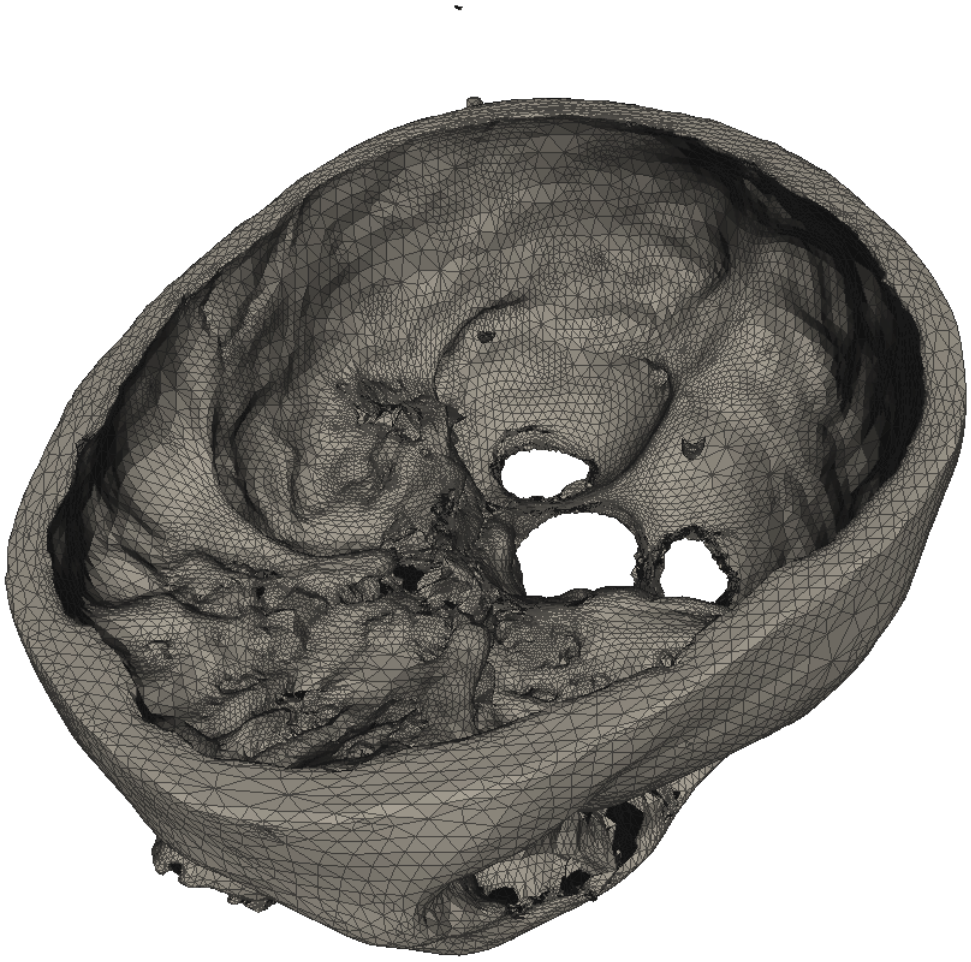
Obrázek A.4: Originální model



Obrázek A.5: Marching Cubes



Obrázek A.6: Hierarchical Iso-Surface Extraction



Obrázek A.7: Hierarchical Iso-Surface Extraction