České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Jakub Groll

Studijní program: Otevřená informatika
Obor: Počítačová grafika a interakce

Název tématu: Dvojúrovňové vykreslování volumetrických dat

Pokyny pro vypracování:

Seznamte se s metodou přímého vykreslování volumetrických dat, která umožňuje kombinovat různé vizualizační techniky při akumulaci barvy a průhlednosti podél paprsku. Dále se seznamte s metodami pro nefotorealistické vykreslování volumetrických dat a implementujte alespoň 2 takové metody. Umožněte alespoň 2 různé kombinace těchto metod při akumulaci barvy a průhlednosti podél paprsku pro segmentovaná data. Svou implementaci ověřte na alespoň 4 různých vstupních datech. Demonstrujte, jakých efektů lze dosáhnout pomocí různých technik a jejich různých kombinací.

Seznam odborné literatury:

1) Engel, Klaus, et al. Real-time volume graphics. CRC Press, 2006.
2) Hauser, Helwig, et al. Two-level volume rendering. IEEE transactions on visualization and computer graphics, 2001, 7.3: 242-252.

Vedoucí: Ing. Ladislav Čmolík, Ph.D.

Platnost zadání: do konce zimního semestru 2018/2019

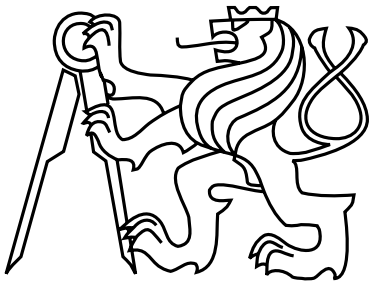
[redacted]
prof. Ing. Jiří Žára, CSc.
vedoucí katedry

L.S.

[redacted]
prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 3.4.2017

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce



Diplomová práce

Dvojúrovňové vykreslování volumetrických dat

Bc. Jakub Groll

Vedoucí práce: Ing. Ladislav Čmolík, Ph.D.

Studijní program: Otevřená informatika, Magisterský

Obor: Počítačová grafika a interakce

24. května 2017

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Ladislavu Čmolíkovi, Ph.D. za odborné vedení práce a cenné rady, které mi pomohly při zpracování této práce. Velký dík patří mé rodině za morální podporu během celého studia, které si velmi vážím.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 24. 5. 2017 .....

Abstract

This thesis deals with a method of rendering volumetric data, which allows us to combine various visualization techniques while accumulating color and transparency along the ray. The aim of this work is to introduce the reader to the volume rendering on GPU and to show how two-level volume rendering can be used on segmented data. Based on the analysis and suggested design of the solution, an implementation is made. This allows us to compose two newely implemented methods of illustrative rendering with existing methods of direct volume rendering. The resulting vizualization application was tested on six volumetric models with grids of different resolutions, demonstrating what effects can be achieved using different techniques and their various combinations. This provides an insight into the behavior of two-level volume rendering under varied conditions.

keywords: volumetric data, ray tracing, transfer function, two-level volume rendering, non-realistic methods, segmented data

Abstrakt

Tato diplomová práce se zabývá metodou vykreslování volumetrických dat, která umožňuje kombinovat různé vizualizační techniky při akumulaci barvy a průhlednosti podél paprsku. Cílem práce je seznámit čtenáře s mechanismem vykreslování volumetrických dat na grafickém procesoru a ukázat, jak lze dvojúrovňové vykreslování aplikovat na segmentovaná data. Na základě analýzy a návrhu řešení je provedena implementace, která umožňuje komponovat dvě nově realizované metody ilustrativního vykreslování se stávajícími metodami přímého vykreslování. Výsledná vizualizační aplikace byla otestována na šesti volumetrických modelech s mřížkami různých rozlišení, kde demonstruji, jakých efektů lze dosáhnout pomocí různých technik a jejich různých kombinací. Tím je umožněn vhled do chování dvojúrovňového vykreslování volumetrických dat za různých podmínek.

klíčová slova: volumetrická data, sledování paprsku, transfer funkce, dvojúrovňové vykreslování, nefotorealistické metody, segmentovaná data

Obsah

Seznam obrázků

Seznam kódů

Kapitola 1

Úvod

Ilustrativní vykreslování volumetrických dat často využívá nefotorealistické techniky k potlačení druhotných struktur volumetrického modelu a ke zviditelnění důležitých částí či objektů zájmu. Tyto techniky těží ze zkušenosti ilustrátorů, kteří se dlouhá staletí potýkali s problémem zobrazení složitých struktur snadno srozumitelnou cestou. Kombinace takových technik, společně s integrováním informace o segmentaci dat do jednoho frameworku není lehký úkol, kvůli značným rozdílům jednotlivých přístupů a jejich parametrů. V této diplomové práci analyzuji metodu *dvojúrovňového vykreslování* [22], která umožňuje kombinace nefotorealistických technik a metod přímého vykreslení dat. Implementace zmíněné metody je realizována v rámci vizualizační aplikace postavené na knihovně Tiger, která dosud zobrazovala volumetrická data klasicky přímou metodou. K ověření funkčnosti takového vykreslování jsem dále implementoval dvě nefotorealistické metody, a to ilustrativní vykreslování dat zachovávající kontext a vykreslování volumetrických dat řízené důležitostí.

Cíl této práce je prozkoumat metodu dvojúrovňového vykreslování a demonstrovat, jakých efektů lze dosáhnout pomocí různých technik a jejich různých kombinací. Tím umožním vhled do chování dvojúrovňového vykreslování volumetrických dat za různých podmínek.

1.1 Struktura práce

Práce je rozdělená do pěti kapitol. V kapitole 2 se nachází nezbytný teoretický základ, kde popisuji volumetrická a segmentovaná data, analyzuji princip volumetrického vykreslování a s ním spjatá kompoziční schémata, aplikace transfer funkcí. V této kapitole se dále věnuji dvěma ilustrativním technikám vykreslování a dvojúrovňovému vykreslování, které tyto a další techniky kombinuje při akumulaci barvy a neprůhlednosti podél paprsku. V závěru kapitoly vysvětluji princip vykreslení dat po hardwarové stránce a navrhuji postup, který následně realizuji v implementační části. Kapitola 3 nahlíží na věc z praktické stránky. Jsou zde představeny technologie, které jsem použil v implementaci. Dále zde popisuji strukturu projektu, tedy z kterých částí se skládá a naopak, které části již byly implementovány v předchozím vývoji. V kapitole 4 diskutuji výsledky testování a závěr práce se nachází v poslední kapitole 5.

Kapitola 2

Analýza a návrh řešení

V následující kapitole analyzuji součásti potřebné pro metodu přímého vykreslování volumetrických dat, která umožňuje kombinovat různé vizualizační techniky při akumulaci barvy a průhlednosti podél paprsku. Úvodní část (sekce 2.3, 2.5, 2.6, 2.9 and 2.10) vychází z bakalářské práce [19]. Tato analýza byla prohloubena a rozšířena o nezbytné úseky potřebné k ustanovení robustního základu práce. Na toto navazuji v sekci 2.11, kde vysvětluji princip dvojúrovňového vykreslování segmentovaných dat. Následující sekce 2.12 a 2.13 rozebírají vizualizační techniky, které jsem na základě literatury analyzoval a později implementoval. Oddíl 2.14 věnuji hardwarové stránce, kdy vysvětlím události odehrávající se na grafické kartě. Hlavní roli zde hraje zobrazovací řetězec (viz 2.14.1). Závěrem kapitoly se zaměřím na návrh řešení, jež nás přiblíží ke kapitole Implementace a upřesní směr, jakým se hodlám vydat na základě analýzy.

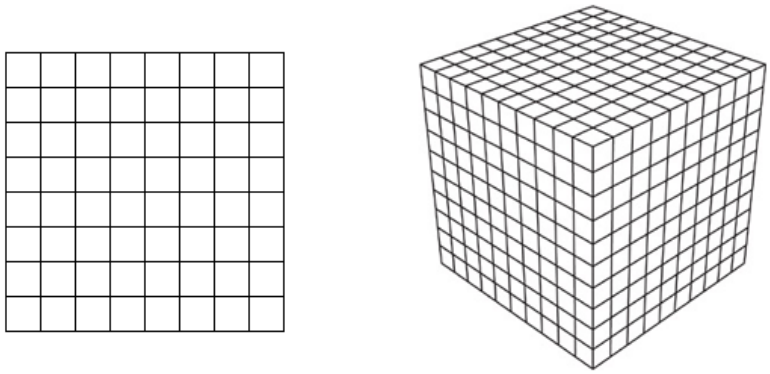
2.1 Volumetrická data

Volumetrická data jsou reprezentována podobným způsobem jako 2D obrázek [21]. Ten se skládá z *pixelů* (zkráceně z angl. "picture elements"), které jsou uspořádány v pravidelné mřížce. Pixely jsou datové prvky 2D obrázku nesoucí informaci o barvě.

Přechodem do vyšší dimenze, tedy z dvojrozměrného prostoru do prostoru trojrozměrného jsou 2D pixely rozšířeny na 3D *voxely*. Voxely (z angl. "volume element") jsou organizovány v pravidelné 3D mřížce tvořící volumetrický objekt. Termín *voxel* má v literatuře dva mírně odlišné významy [21]. V této práci bude brán voxel jako bod 3D prostoru společně s interpolační funkcí, která vyplňuje prostor mezi takovými vzorky. Voxely slouží jako vrcholy uniformní mřížky. které jsou spojeny hranami a tvoří šestistěnné buňky. Obrázek 2.1 ilustruje uniformní mřížky ve dvou a třech dimenzích.

2.1.1 Zdroje dat

Volumetrická data mohou být získána a aplikována v mnoha vědních oborech [21]. Důležitým typem aplikace je vizualizace vědeckých dat, konkrétně dat medicínských. V lékařském



Obrázek 2.1: Příklady uniformních mřížek: 2D uniformní mřížka (vlevo) a 3D uniformní mřížka (vpravo).

zobrazování jsou data typicky získána snímajícím zařízením.

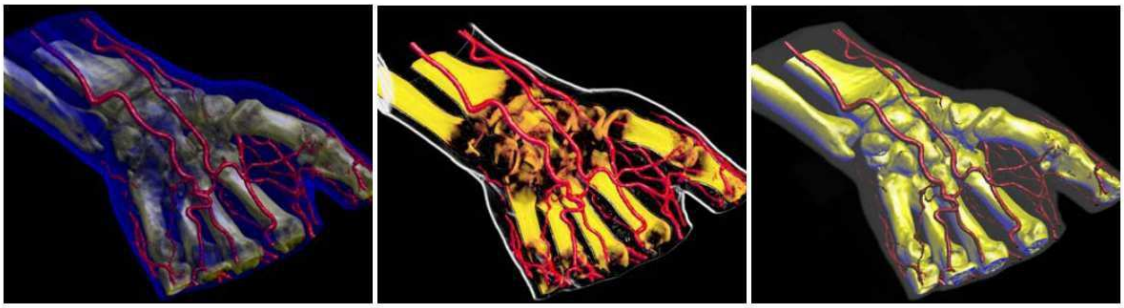
CT (výpočetní tomografie) je běžně užívaná metoda k získání medicínských 3D dat. Proces skenování je založen na rentgenovém záření. Paprsky takového záření jsou vyzařované z jedné strany a záření, které prochází pacientovým tělem je zaznamenáno na druhé straně. Zářič rotuje okolo pacientova těla a získává snímky utlumení radiace z různých směrů. Z těchto snímků je zrekonstruována výsledná 3D mřížka skalárních dat.

MR či MRI (magnetická rezonance) využívá velké magnetické pole a elektromagnetické vlnění s vysokou frekvencí. Na rozdíl od CT vyšetření, které je s MR někdy alternativní, tedy nenese žádná rizika způsobená zářením (nulová radiační zátěž). Nevýhodou vyšetření MR je určitá hlučnost zařízení. Podstatou barevného odlišení jednotlivých tkání je jejich rozdílné chování při stejném vnějším působení. Vyšetření se provádí buď s nebo bez použití kontrastní látky [4].

Další způsoby získání volumetrických dat jsou popsány v literatuře [21].

2.2 Segmentovaná data

Významným cílem, a to zejména při práci s medicínskými daty, je možnost vizuálně rozlišit některé konkrétní objekty v souboru dat, jež jsou předmětem zájmu. Tyto objekty by pak nejen šlo rozlišit, ale také zcela zanedbat při vykreslování, což by umožnilo vyobrazení jiných struktur. *Segmentace* je proces identifikování či označení jednotlivých voxelů, jakožto příslušnosti, k již zmíněným objektům tvořící objem dat [21]. Obrázek 2.2 ukazuje příklad využití segmentace na snímcích medicínských dat, konkrétněji CT skenu lidské ruky. Zde identifikujeme čtyři objekty: strukturu kostí, cévní soustavu, kůži a vzduch obklopující ruku. První tři objekty byly vykresleny pomocí rozdílných optických vlastností a voxely reprezentující vzduch byly zcela vynechány neboli oříznuty [21].



Obrázek 2.2: Segmentovaný soubor dat zápěstí (256×128×256) se třemi objekty: kůže, cévní soustavy a struktury kostí. Dvojúrovňové vykreslování (viz. 2.11) obsahuje různé transfer funkce, vykreslení a kompoziční schémata: (vlevo) všechny objekty vykresleny pomocí přímého vykreslování dat se stínováním; kůže částečně zakrývá kostní strukturu; (uprostřed) kůže vykreslena pomocí nefotorealistické metody vykreslující obrysy složenou MIP kompozičním schématem, kosti vykresleny přímou metodou, cévy jsou tónovány; (vpravo) kůže je vykreslena pomocí MIP, kosti jsou tónované a na cévní soustavu byla aplikována metoda iso-surfacing. Zdroj [20].

V mnohých vykreslovacích metodách jsou všechny voxely jednoho objemového souboru dat brány totožným způsobem - tedy bez využití informace o příslušnosti voxelů k objektům. V tomto případě je vizuálního rozlišení obvykle dosaženo díky transfer funkcím. Multidimenzionální transfer funkce [27, 29] se jeví jako silný nástroj pro usnadnění vnímání různých objektů. V posledních letech byly poté nefotorealistické přístupy [43, 10, 35] také úspěšně využity ke zlepšení vnímání nesegmentovaných objektů. Vykreslovací techniky využívající zmíněných nástrojů ovšem často nestačí k rozlišení objektů dle specifických potřeb uživatele. S tímto problémem se potýká již zmíněná segmentace, při které vytvoříme informaci o příslušnosti k objektům (angl. object membership information). Zde bych chtěl zmínit knihu [49], kde jsou velmi dopodrobna analyzovány principy a techniky segmentace nejen z lékařského pohledu.

Zapracování informace o segmentaci společně s více vykreslovacími metodami a různými množinami parametrů do kvalitního rendereru ovšem není triviální úkol. Dalším úskalím jsou pak omezené možnosti hardwaru, kde se snažíme například minimalizovat zátěž na paměť využitím pouze jedné *segmentační masky* (viz. dále). Grafický hardware také nedokáže snadno interpolovat mezi voxely patřícími odlišným objektům, ovšem použití segmentace bez takového filtrování dává vzniknout nechtěným artefaktům. Tím vzniká další problém správného filtrování hranic objektů k získání vysoce kvalitních snímků s konzistentním přiřazováním fragmentů ovšem bez vzniku neexistujících ID objektů [20]. Na vykreslování segmentovaných dat se blíže podíváme v sekci 2.11.

2.3 Přímé vykreslování volumetrických dat

Předtím, než objasním princip vykreslování pomocí ilustrativních metod, je třeba si vysvětlit základní přístup pro výpočet barvy a neprůhlednosti volumetrických dat - přímé vykreslování volumetrických dat. Velká část této kapitoly je věnována pojmu přímého vykreslování, který si vysvětlíme, přičemž analyzujeme jeho součásti a metody, ze kterých se skládá.

Dle [36] přímé vykreslování dat označuje techniky, při kterých se vykresluje výsledný obraz *přímo* z volumetrických dat. Nevyužívá se zde mezivýsledků či přechodných konstruktů jako například u vytváření kontur povrchů (angl. Contour surface polygons). Tyto techniky vyžadují optický model chování dat při různých situacích působení světla. V sekci 2.5 takové modely představím a popíši. Jak již bylo zmíněno, hlavní myšlenkou metody přímého vykreslování je získat 2D obraz z dat přímo. Přesněji se snažíme vizualizovat extrahované informace z trojrozměrného pole skalárních hodnot, což může být zapsáno jako zobrazení

$$\phi = \mathbb{R}^3 \mapsto \mathbb{R}, \tag{2.1}$$

neboli se jedná o funkci ze 3D prostoru do skalární hodnoty [21]. U těchto dat předpokládáme, že reprezentují semi-transparentní objekt vyzařující (odrážející) světlo. Přímé vykreslování volumetrických dat je výpočetně velmi náročné a může být provedeno několika postupy (viz sekce 2.6).

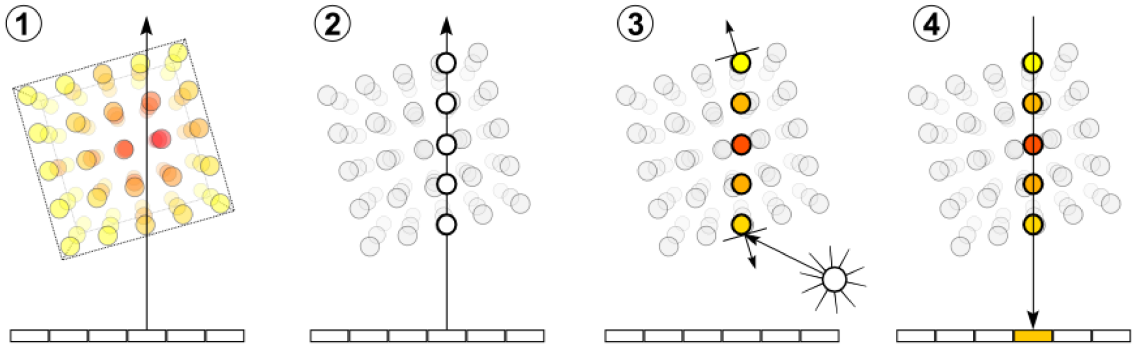
Rychlost vykreslování byla vždy závažným problémem u přímého vykreslování volumetrických dat, jelikož všechny části objemu mohou přispívat k výslednému obrázku a zároveň nová orientace scény obvykle vyžaduje značné přepočítávání [17]. Spektrum metod nabízejících rozdílné kombinace rychlostí a kvality obrázků bylo představeno v publikacích: [13, 45, 50, 33, 30, 37]. Image-order algoritmy, jako například sledování paprsku (ray casting), produkují lepší kvalitu obrázků, ovšem na úkor rychlosti, kde hlavním faktorem je redundantní procházení struktury dat [34, 11]. Některé object-order metody využívají hardwarovou podporu, založenou na Goraudově stínování, při které zrychlují vykreslování tím, že vypočítávají projekce objektů ve scéně a následně s nimi pracují jako s polygony [46, 32]. Podrobněji jsou jednotlivé metody kategorizovány a popsány v sekci 2.10.

2.4 Sledování paprsku

Dle [21], sledování paprsku neboli ray casting je nejvíce používaná Image-order 2.10 technika pro vykreslování volumetrických dat. Základní myšlenkou této metody je přímo vypočítávat integrál pro vykreslování volumetrických dat 2.10 podél všech paprsků, které jsou vrženy z kamery. Pro každý pixel ve scéně je určen právě jeden paprsek. Poté vzorkujeme v určitých intervalech objemová data našeho objektu. Princip sledování paprsku je znázorněn na Obrázku 2.3. Zde vidíme, že nejprve je vyslán paprsek pro každý pixel ve scéně (1) a následně vzorkujeme objekt v daných intervalech (2). Volitelně aplikujeme transfer funkci (viz. 2.9) (3) a nakonec vypočítáme výslednou barvu a neprůhlednost pomocí diskretizovaného integrálu pro vykreslení volumetrických dat (4). Tento pojem bude vysvětlen v následující

sekci.

Pro techniku sledování paprsku bylo vyvinuto několik způsobů, jak zrychlit vykreslovací proces. Jedná se například o *Early Ray Termination* či *Space Leaping*. Více se lze dočíst v publikaci [44].



Obrázek 2.3: Ray casting - vyslán paprsek pro každý pixel ve scéně (1), Vzorkování (2), Interpolování hodnot a aplikace transfer funkce (3), Iterativní výpočet diskretizovaného integrálu pro vykreslení volumetrických dat (4). Zdroj [5].

2.5 Vykreslovací rovnice

Následující dvě sekce (2.5 a 2.6) čerpají kvůli své komplexnosti z knihy Real-time volume graphics [21], která je díky svým autorům: Markus Hadwiger, Joe M. Kniss, Christof Rezk-salama, Daniel Weiskopf a Klaus Engel, jedním z nejkompetentnějších zdrojů dané problematiky. Účelem této kapitoly je popsat a vysvětlit některé z fyzikálních a matematických vlastností rovnic vyskytujících se ve výpočtu přenosu světla při vykreslování volumetrických dat.

2.5.1 Teorie přenosu světla

Fyzikální základ pro vykreslování volumetrických dat se opírá o geometrickou optiku, kde se předpokládá, že se světlo šíří prostorem přímočaře, vyjma situací, kdy se světlo střetne s jinými objekty (angl. participating media). Tudíž vzájemné působení mezi světlem a hmotou je situace, které se budeme nyní věnovat. Následující typy interakcí paprsku světla s hmotou jsou brány v úvahu:

Vyzařování (angl. Emission)
Některé materiály (například plyny) vyzařují světlo, převodem tepla, či jiných vlastností, na energii.

Absorpce (angl. Absorption)
Různé materiály mohou naopak pohlcovat (absorbovat) světlo, které je následně převedeno v teplo. Energie generující světlo je v tomto případě redukována.

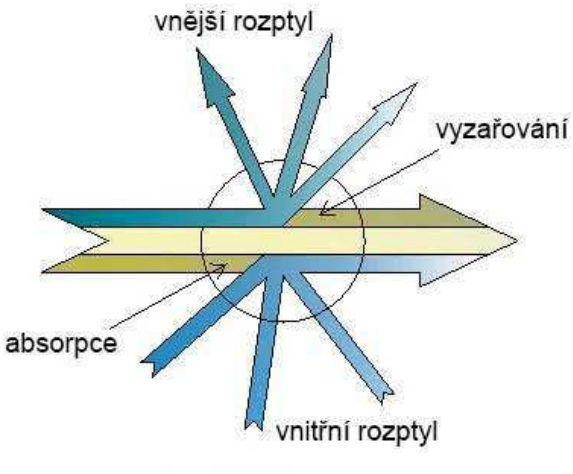
Rozptyl (angl. Scattering)
Světlo se přirozeně odráží a rozptyluje od ostatních objektů. V této situaci se mění směr šíření světla, jak můžeme vidět na Obrázku 2.4.

Vyzařování, absorpce a rozptyl ovlivňují množství radiační energie podél světelného paprsku. Energie světla může být popsána pomocí svojí *záře* I (angl. *radiance*), která je definována jako radiační energie Q na jednotku plochy, na úhel Ω a jednotku času t:

$$I = \frac{\mathrm{d}Q}{dA_{\perp}\mathrm{d}\Omega\,\mathrm{d}t}. \tag{2.2}$$

Index $\perp$ označuje skutečnost, že plocha je měřena podél paprsku světla: $A_{\perp} = A\cos\theta$, kde θ je úhel mezi směrem světelného paprsku a normálovým vektorem povrchu A.

Zář je fundamentální veličinou pro počítačovou grafiku, jelikož se nemění její hodnota podél paprsku ve vakuu. Další informace lze získat z knihy [18], či [23]. Dále jsem přejal českou terminologii z přednášek předmětu APG [2].



Obrázek 2.4: Typy vzájemného působení světla s objektem. Upraveno z [55].

Přítomnost objektu interagujícího se světlem ovlivňuje záři podél paprsku (viz Obrázek 2.4). Absorpce redukuje energii světla, naopak vyzařování ji zvyšuje. Rozptyl může, jak redukovat, tak zvyšovat energii. Kombinací absorpce, vyzařování a rozptylu získáme následující rovnici přenosu světla:

$$\omega.\nabla_{\mathbf{x}} I(\mathbf{x},\omega) = -\chi I(\mathbf{x},\omega) + \eta \tag{2.3}$$

Výraz $\omega.\nabla_{\mathbf{x}} I$ je skalární součin mezi vektorem směru světla ω a gradientu záře I vzhledem k pozici $\mathbf{x}$. Operátor ∇ je kratší notace pro $\nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z)$. χ dále značí koeficient celkové absorpce, jenž určuje míru zeslabení světla objektem. Veličina η je celkové vyzařování a udává míru zvýšení radiační energie pomocí objektu.

Koeficient celkové absorpce se skládá ze dvou částí: pravého absorpčního koeficientu κ (angl. true absorbtion coefficient) a koeficientu rozptylu σ (angl. scattering coefficient), který reprezentuje ztrátu energie z rozptylu. Koeficient celkové absorpce pak můžeme psát jako

$$\chi = \kappa + \sigma. \tag{2.4}$$

Analogicky, koeficient celkového vyzařování můžeme rozdělit na složku q reprezentující vyzařování (např. termální excitace) a složku rozptylu j:

$$\eta = q + j. \tag{2.5}$$

Ačkoliv veličiny závisí na pozici $\mathbf{x}$ a směru ω, pro přehlednost rovnic je vynechávám. Dále bych poznamenal, že κ, σ a q se často přiřazují na základě transfer funkcí (viz. 2.9), jelikož se jedná o optické vlastnosti materiálu. Tento fakt ovšem neplatí pro koeficient rozptylu j, který musí být nepřímo vypočítán dle vlastností materiálu objektu. Přesněji, je spočten z příspěvků všech směrů příchozího světla, což vede na rovnici:

$$j(\mathbf{x}, \omega) = \frac{1}{4\pi} \int_{koule} \sigma(\mathbf{x}, \omega') p(\mathbf{x}, \omega', \omega) I(\mathbf{x}, \omega') \, \mathrm{d}\Omega'. \tag{2.6}$$

Zde se příspěvky z dopadajícího světla $I(\mathbf{x}, \omega')$ kumulují integrováním přes všechny směry ω'. Dále jsou tyto příspěvky ovlivněny koeficientem rozptylu σ a *fázovou funkcí* p, která popisuje pravděpodobnost, že je světlo rozptýleno z původního směru ω' do nového směru ω. Fázová funkce tedy reprezentuje rozptýlení světla v závislosti na úhlu. Rozdílné materiály mohou mít různé fázové funkce, které mohou vést k odlišným vzhledům objemu. Mělo by být zmíněno, že zde není hlubší význam ve zlomku $1/4\pi$. Je zde jednoduše proto, aby vykrátil faktor 4π, který vznikne integrováním fázové funkce přes celou kouli.

Zpracováním výše uvedených situací získáme kompletní rovnici pro výpočet šíření světla:

$$\omega \cdot \nabla_{\mathbf{x}} I(\mathbf{x}, \omega) = -(\kappa(\mathbf{x}, \omega) + \sigma(\mathbf{x}, \omega)) I(\mathbf{x}, \omega) + q(\mathbf{x}, \omega) + \int_{koule} \sigma(\mathbf{x}, \omega') p(\mathbf{x}, \omega', \omega) I(\mathbf{x}, \omega') \, \mathrm{d}\Omega', \tag{2.7}$$

kde proměnné mají následující význam: $\mathbf{x}$ - pozice a ω' - směr původního paprsku, ω - směr paprsku odraženého, dále κ - koeficient absorpce, I - zář, σ - koeficient rozptylu, p - fázová funkce, q - koeficient vyzařování.

Dosud jsme byli schopni popisovat pomocí záře I pouze obrázky ve stupních šedi. K tomu, abychom umožnili vykreslování barevných obrázků, musíme pracovat s vlnovými délkami záření, typicky skrze záři závislou na vlnových délkách $I_\lambda = \mathrm{d}I/\mathrm{d}\lambda$. Ve většině případů se nepočítá se změnou ve vlnových délkách (např. kvůli neelastickému rozptylování), a tudíž se Rovnice 2.7 dá vyřešit pro každou vlnovou délku zvlášť. Předpokládáme-li, že pracujeme pouze s elastickým rozptylem, pak jsou barevné obrázky obvykle počítány pouze pro několik pásem vlnových délek (např. červená, zelená a modrá).

2.5.2 Optické modely

Jelikož výpočet komplikované rovnice 2.7 je příliš složitý, často se využívá zjednodušených modelů. U těchto modelů se vynechají některé prvky z množiny interakcí vyjmenovaných v podsekci 2.5.1. Tímto přístupem získáme rovnice, které jsou lépe uchopitelné. Přehled modelů vypadá následovně:

Absorption Only Model. Nejjednodušší model. Objem tělesa je tvořen chladným, dokonale černým materiálem, který může absorbovat dopadající světlo. Žádné světlo ovšem není vyzařováno či rozptylováno do okolí.

Emission Only Model. Objem tělesa je tvořen plynem, který vyzařuje světlo, ale je zcela průhledný. Absorpce a rozptyl jsou zanedbány.

Emission-Absorption Model. Tento optický model je nejznámějším modelem u vykreslování volumetrických dat. Materiál může vyzařovat světlo a pohlcovat dopadající světlo. Rozptyl a nepřímé osvětlení zanedbáváme.

Single Scattering and Shadowing Model. Tento model obsahuje takový rozptyl, jenž přichází z externího zdroje světla. Obsahuje stíny, které berou v potaz dopadající světlo z externího zdroje.

Multiple Scattering Model. Zde je cílem vypočítat kompletní osvětlovací model pro objemová data. Počítá s absorpcí, vyzařováním i rozptylem.

Velmi podrobný popis jednotlivých modelů obsahuje článek [36]. Já zde přiblížím pouze jeden z modelů, který pro nás hraje podstatnou roli. Emission-Absorption Model je nejvíce používaným modelem pro vykreslování volumetrických dat, jelikož se jedná o kompromis mezi efektivitou výpočtu a faktu, že je rovnice přenosu světla dostatečně obecná. Tento model, počítající s vyzařováním a absorpcí světla, avšak ignorující rozptylování a nepřímé

osvětlení, vede na následující rovnici

$$\omega \cdot \nabla_{\mathbf{x}} I(\mathbf{x}, \omega) = -\kappa(\mathbf{x}, \omega) I(\mathbf{x}, \omega) + q(\mathbf{x}, \omega), \tag{2.8}$$

kde veličiny mají stejný význam jako v Rovnici 2.7. Tato rovnice je označována jako *vykreslovací rovnice* (angl. volume-rendering equation). Přesněji se jedná o vykreslovací rovnici v diferenciální formě, jelikož popisuje šíření světla změnou jeho vyzařování. Pokud uvažujeme šíření jediného paprsku světla, může být rovnice 2.8 přepsána jako

$$\frac{\mathrm{d}I(s)}{\mathrm{d}s} = -\kappa(s) I(s) + q(s), \tag{2.9}$$

kde jednotlivé pozice jsou popsány délkou s měřenou od počátku vyslaného paprsku. Ostatní veličiny mají význam jako v předchozím textu, neboli κ - koeficient absorpce, I - záře, q - koeficient vyzařování.

2.5.3 Integrál pro vykreslování volumetrických dat

Vykreslovací rovnice (Rovnice 2.9) může být řešena pro záři integrováním podle směru světelného paprsku od začátku $s = s_0$ až po konec paprsku $s = D$, což vede na *integrál pro vykreslování volumetrických dat* (angl. volume-rendering integral)

$$I(D) = I_0 e^{-\int_{s_0}^{D} \kappa(t)\,\mathrm{d}t} + \int_{s_0}^{D} q(s) e^{-\int_{s_0}^{D} \kappa(t)\,\mathrm{d}t}\,\mathrm{d}s. \tag{2.10}$$

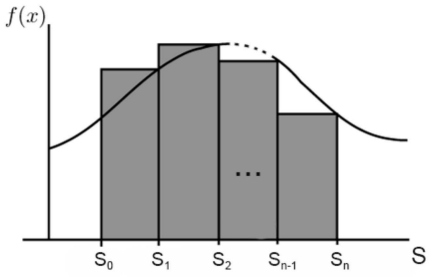
Zde I_0 představuje světlo vstupující do objemu tělesa z pozadí na pozici $s = s_0$; $I(D)$ představuje zář opouštějící objem na pozici $s = D$, která následně vstupuje do kamery. První člen v Rovnici 2.10 popisuje světlo z pozadí ovlivněné objektem ve scéně. Člen

$$\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(t)\,\mathrm{d}t \tag{2.11}$$

je definován jako *optická hloubka* mezi pozicemi s_1 a s_2. Optická hloubka má fyzikální význam, určující jak dlouho světlo může letět, než je absorbováno. Malé hodnoty indikují, že objekt je více průhledný, naopak větší hodnoty udávají objekt spíše neprůhledný. Pro doplnění κ je koeficient absorpce a q koeficient vyzařování.

2.5.4 Diskretizace

Hlavním cílem vykreslování volumetrických dat je výpočet integrálu 2.10. Jelikož tento výpočet ve většině případů nelze vyřešit analyticky, na řadu přicházejí numerické metody, které se snaží najít aproximace původního integrálu, známé pod pojmem diskretizace[1].



Obrázek 2.5: Aproximace integrálu - Riemannův přístup. Zdroj [7].

Běžný způsob je rozdělení integrálu na malé intervaly, které jsou popsány souřadnicemi $s_0 < s_1 < ... < s_{n-1} < s_n$, se začátkem v s_0 a koncem v $s_n = D$ (viz Obrázek 2.5).

Je třeba zmínit, že není podmínkou, aby intervaly měly stejné délky. S tímto faktem můžeme získat hodnotu záře na pozici s_i

$$I(s_i) = I(s_{i-1})T(s_{i-1}, s_i) + \int_{s_{i-1}}^{s_i} q(s)T(s, s_i)\,\mathrm{d}s. \tag{2.12}$$

Zde zavádíme značení T - průhlednost a c_i - barva (neboli příspěvek záře) i-tého intervalu.

$$T_i = T(s_{i-1}, s_i), \quad c_i = \int_{s_{i-1}}^{s_i} q(s)T(s, s_i)\,\mathrm{d}s. \tag{2.13}$$

Hodnota záře, v místě kde paprsek opouští objekt, je

$$I(D) = I(s_n) = I(s_{n-1})T_n + c_n = (I(s_{n-2})T_{n-1} + c_{n-1})T_n + c_n = ...,$$

což lze přepsat jako

$$I(D) = \sum_{i=0}^{n} c_i \prod_{j-i+1}^{n} T_j, \quad \text{kde } c_0 = I(s_0). \tag{2.14}$$

[1] diskretizace, fyz. náhrada spojitého prostředí (kontinua) systémem diskrétních bodů

Nyní se nacházíme ve stavu, kdy máme integrál rozdělen na n intervalů, a násobky a součty v Rovnici 2.14 lze spočítat. Chybějící částí je tedy jen vyhodnocení příspěvků průhlednosti a barvy jednotlivých intervalů.

Nejběžnější způsob, kterým tohoto lze dosáhnout, je aproximovat integrál pro vykreslení volumetrických dat v Riemannově smyslu. To znamená integrovat přes n stejně vzdálených segmentů délky $\Delta x = (D - s_0)/n$. Zde je funkce aproximována po částech konstantní funkcí, jak je to znázorněno na Obrázku 2.5. V této aproximaci je průhlednost i-tého segmentu

$$T_i \approx e^{-\kappa(s_i)\Delta x} \tag{2.15}$$

a příspěvek barvy i-tého segmentu odpovídá

$$c_i \approx q(s_i)\Delta x, \tag{2.16}$$

kde κ je koeficient absorpce a q koeficient vyzařování.

2.6 Kompoziční schémata

Kompoziční schémata (angl. Compositing schemes) jsou základem pro iterativní přístup výpočtu diskretizovaného integrálu pro vykreslení volumetrických dat (Rovnice 2.14).

Rozlišujeme dva základní přístupy - vykreslování zepředu dozadu (dále jen angl. front-to-back) a vykreslování zezadu dopředu (dále angl. back-to-front). Myšlenkou těchto přístupů je sekvenční zpracování výše zmíněné rovnice.

2.6.1 Front-to-back

Je-li použito front-to-back schéma, putují zobrazovací paprsky od kamery k objektu. Iterativní formulace rovnice vyjadřující tuto metodu začíná na pozici nejblíže kameře a končí na zadní straně našeho objektu. Někdy je tato metoda nazývána jako *Under operator* [15].

V následujících rovnicích bude proměnná C reprezentovat barvu (typicky RGB hodnotu). Oba příspěvky z aktuální hodnoty záře c a akumulované hodnoty záře I, s kterými jsme se setkali v předchozích sekcích, jsou nyní spjaty s touto proměnnou. T reprezentuje průhlednost. Rovnice vypadají následovně:

$$\hat{C}_i = \hat{C}_{i+1} + \hat{T}_{i+1}\hat{C}_i,$$
$$\hat{T}_i = \hat{T}_{i+1}(1 - \alpha_i),$$

s počátečními hodnotami[2]

[2] v tomto případě je první pozice $i = n$ nejblíže kameře

$$\hat{C}_n = C_n,$$
$$\hat{T}_n = 1 - \alpha_n.$$

Výsledkem aktuálního iteračního kroku jsou $\hat{C}_i$ a $\hat{T}_i$. $\hat{C}_{i+1}$ a $\hat{T}_{i+1}$ jsou akumulované hodnoty předchozích výpočtů. C_i a α_i jsou dány transfer funkcí (viz 2.9) nebo pochází ze samotného fyzikálního modelu látky.

Přepsáním proměnných $C_{dst} = \hat{C}_j$ $(j = i,i+1)$, $C_{src} = C_i$, $\alpha_{dst} = 1 - \hat{T}_j(j = i, i+1)$, $\alpha_{src} = \alpha_i$ získáme častější zápis rovnice:

$$\begin{aligned} C_{dst} &\leftarrow C_{dst} + (1 - \alpha_{dst})C_{src}, \\ \alpha_{dst} &\leftarrow \alpha_{dst} + (1 - \alpha_{dst})\alpha_{src}. \end{aligned} \tag{2.17}$$

Zde C reprezentuje barvu a α neprůhlednost. Proměnné s dolním indexem $_{src}$ popisují vstupní hodnoty. Naopak proměnné s dolním indexem $_{dst}$ popisují výstupní hodnoty, které akumulují hodnoty barvy a neprůhlednosti.

2.6.2 Back-to-front

Otočením směru procházení scény dle paprsku získáme back-to-front schéma, dle [15] označované jako *Over operator*:

$$\hat{C}_i = \hat{C}_{i-1}(1 - \alpha_i) + C_i,$$
$$\hat{T}_i = \hat{T}_{i-1}(1 - \alpha_i),$$

s počátečními hodnotami[3]

$$\hat{C}_0 = C_0,$$
$$\hat{T}_0 = 1 - \alpha_0.$$

Poznamenejme, že u této metody nepotřebujeme znát akumulovanou hodnotu průhlednosti, tím pádem můžeme druhou rovnici vynechat. Iterativní rovnice pak vypadá následovně:

$$C_{dst} \leftarrow (1 - \alpha_{src})C_{dst} + C_{src}. \tag{2.18}$$

Kompozičních schémat existuje velké množství. V této sekci jsem popsal základní přístupy schémat front-to-back a back-to-front. Ke konci kapitoly se budu věnovat dalším kompozičním schématům, které budu implementovat v rámci metod ilustrativního vykreslování, konkrétně podsekce 2.12.1 a 2.13.2.

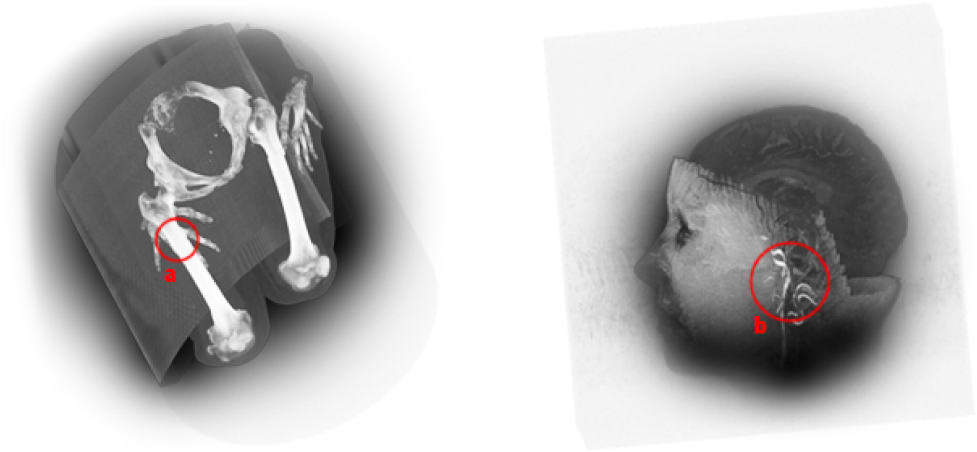
[3] iterace začíná v bodě $i = 0$ a končí v $i = n$

2.7 Maximum intensity projection

Maximum intensity projection je alternativou pro řešení rovnice přenosu světla. Toto schéma je hojně využíváno v medicínském prostředí. Je použitelné zejména na tomografická data po aplikování kontrastních látek (založených například na bázi jódu). Metoda je postavena na výpočtu barvy pixelu, jehož hodnota je určena jako maximum ze všech intenzit měřených podél paprsku:

$$I = \max_{k=0..N} (s_k), \tag{2.19}$$

kde s_k je skalární hodnota měřená podél paprsku, N je počet vzorků a I výsledná intenzita. Velmi názorný je Obrázek 2.6 (vlevo), kde maximální intenzitu v těle představují kosti. Všimněte si, že ztrácíme informaci o objemu s nižší intenzitou za i před těmito vykreslenými částmi. Hlavní aplikací pro MIP je virtuální angiografie - zobrazení cévních struktur v medicínských snímcích (viz článek [42]).



Obrázek 2.6: Maximum Intensity Projection, pánevní oblast těla (vlevo) a MRI sken hlavy (vpravo). Na obrázku je označeno místo, kde ztrácíme informaci o struktuře ruky, která není vykreslena kvůli vyšším hodnotám nacházejících se v oblasti stehenní kosti (a); zobrazení cévních struktur pomocí této metody (b).

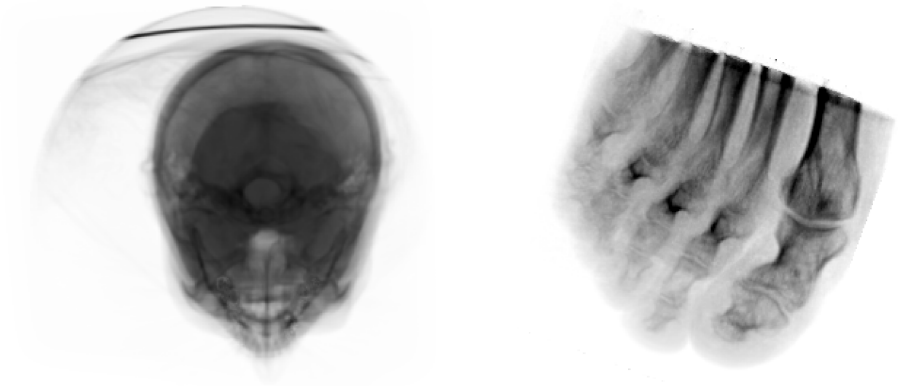
2.8 Average intensity projection

Average intensity projection (viz Obrázek 2.7) je metoda akumulace světla podobná MIP. Rozdíl nastane ve výpočtu barvy pixelu, kde výsledná barva bude vykreslena jako hodnota

průměru intenzit podél paprsku:

$$I = \frac{1}{N} \sum_{k=0}^{N} s_k \tag{2.20}$$

Zde je s_k skalární hodnota měřená podél paprsku, N počet vzorků a I výsledná intenzita.



Obrázek 2.7: Average Intensity Projection s použitím převrácených hodnot barev pro lepší přehlednost, lidská lebka (vlevo) a část chodidla (vpravo)

2.9 Transfer funkce

Pro vyřešení rovnic přenosu světla ze sekce 2.5 musíme znát optické vlastnosti, jako jsou koeficienty vyzařování a absorpce v každém bodě uvnitř objemu tělesa [21]. Až doposud jsme předpokládali, že nám jsou tyto parametry známy. V běžné praxi ovšem dostáváme volumetrické soubory dat, která obsahují abstraktní skalární data. Obecně proto nelze říci, jaké koeficienty vyzařování a absorpce přísluší takovým datům. Namísto toho se uživatel musí rozhodnout, jak by různé struktury měly vypadat přiřazením optických vlastností datům pomocí mapování. Toto mapování je nazýváno *transfer funkce*. Proces nalezení odpovídající transfer funkce se označuje jako *klasifikace* [21].

2.9.1 Klasifikace

Klasifikace je definována jako proces identifikování oblastí zájmu na základě abstraktních hodnot dat [21]. Klasifikace je v zásadě úloha rozpoznávání vzorů z nezpracovaných dat, kde různé vzory přiřazujeme do odlišných kategorií či tříd. Velmi podrobný přehled a shrnutí teorie k tomuto tématu obsahuje kniha [14].

Na transfer funkce je často nahlíženo jako na funkce, jež vezmou definiční obor vstupních hodnot dat a převedou je na rozsah RGBA. Transfer funkce jsou ovšem také využívány k identifikování určitých vzorů a následnému přiřazení rozsahu hodnot zdrojových dat, jenž odpovídá našemu objektu zájmu. Navrhování a vytváření transfer funkcí je manuální, zdlouhavá a časově náročná práce, která vyžaduje výbornou znalost struktur přítomných v datovém souboru [21]. Rád bych zde zmínil článek [27] a z něj vycházející diplomovou práci zabývající se automatickým generováním transfer funkcí [28], jež se snaží zrychlit tento proces a v samé podstatě ulehčit přímé vykreslování volumetrických dat.

Dle autora Kniss et al. [29] se přímé vykreslování volumetrických dat prokázalo jako efektivní a flexibilní metoda pro vykreslování 3D skalárních souborů dat. Transfer funkce jsou fundamentální složkou přímého vykreslování dat, jelikož jejich role je v podstatě vykreslování dat na základě optických vlastností (jako jsou barva a průhlednost). Tyto hodnoty poté transfer funkce přiřazuje volumetrickým datům, jež pak mohou být běžným způsobem vykresleny na grafické kartě.

Transfer funkce byly dlouhou dobu omezeny pouze na funkce jednorozměrné 2.9.2, které dané vlastnosti přiřazují volumetrickým datům jen na základě hodnot těchto dat. V mnohých případech se s tímto typem transfer funkcí nedá správně zachytit a vizualizovat míněnou oblast. Poté je třeba použít vícerozměrných transfer funkcí 2.9.3. Zdroj [29].

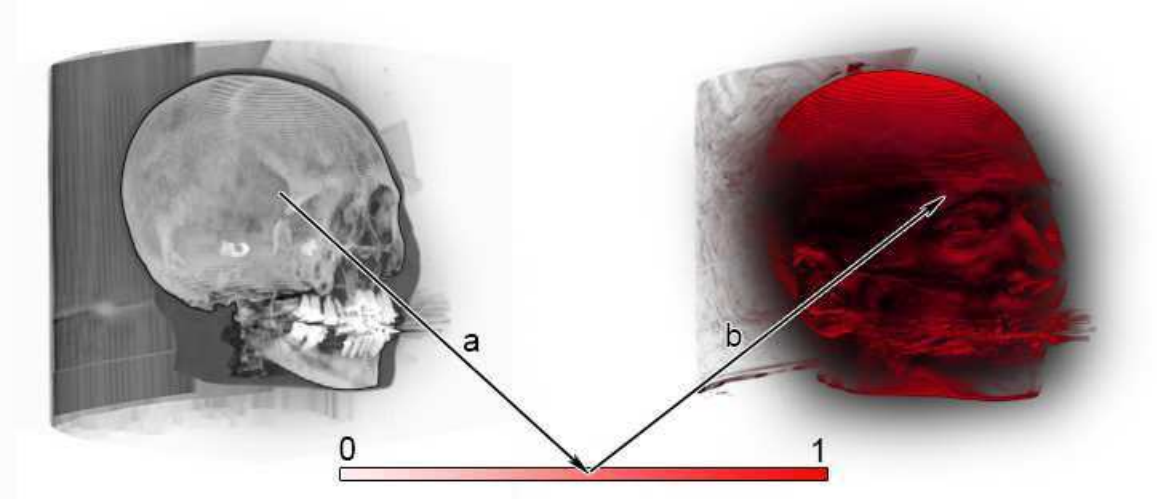
2.9.2 Jednorozměrné transfer funkce

V nejjednodušším případě se transfer funkce mapují na základě hodnoty skalárních dat. Obrázek 2.8 ukazuje, jak funguje mapování barvy na náš objekt. Každá skalární hodnota volumetrických dat je převedena do intervalu $\langle 0, 1 \rangle$, kde 0 odpovídá nejmenší hodnotě skalárních dat objektu a 1 naopak hodnotě největší. Každé hodnotě tedy přísluší určitý vektor kanálů RGBA, který je poté zapsán do fragmentu jako výsledná barva.

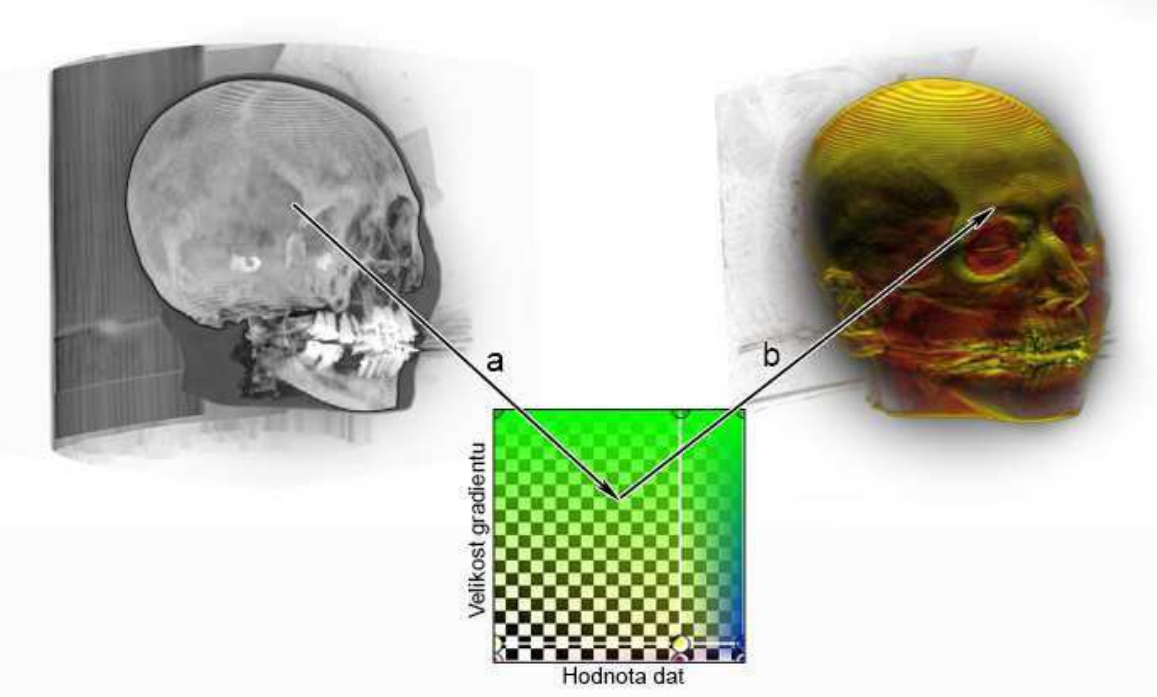
2.9.3 Dvojrozměrné transfer funkce

Přidáním druhého rozměru transfer funkci dostáváme možnost mapovat hodnoty na základě více parametrů. Vhodnou proměnnou se ukázala být hodnota velikosti *gradientu* [29]. Jelikož nám tato veličina udává, jak rychle se hodnoty mění v přilehlém okolí, je ideální pro zvýraznění přechodu mezi objekty s rozdílnou hustotou. Toho se využívá zejména v lékařství pro rozlišení různých druhů tkání [26]. Připomeňme, že existují i transfer funkce s dalšími rozměry - v této diplomové práci ovšem budeme pracovat pouze s 1D a 2D transfer funkcemi.

Na Obrázku 2.9 vidíme transfer funkci, na jejíž svislé poloose určuje hodnotu barvy právě velikost gradientu. Na výsledném snímku se to poté projeví žlutozeleným zbarvením. Tento jev nastává na rozhraní kostí a zubů. Vnitřní struktury jsou pak vykresleny červeně až modře (na obrázku částečně zakryto).



Obrázek 2.8: Použití 1D transfer funkce, převedení hodnot do intervalu $\langle 0, 1 \rangle$ (a), zapsání barvy do fragmentu (b)

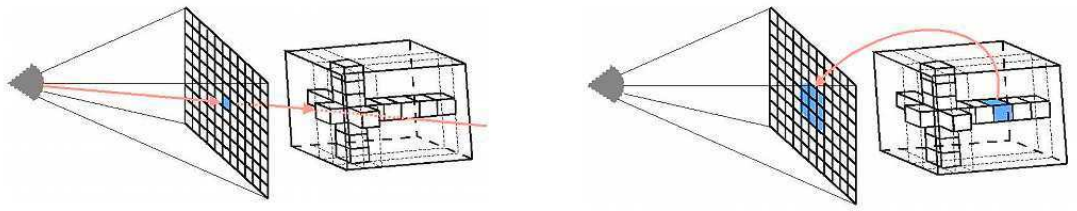


Obrázek 2.9: Použití 2D transfer funkce; 2D transfer funkce (uprostřed) byla aplikována na znázorněná data (vlevo), výsledek použití 2D transfer funkce s veličinou velikosti gradientu (vpravo); Převedení hodnot do intervalu 2D funkce (a), zapsání barvy do fragmentu (b).

2.10 Vykreslovací techniky

Pro schémata popsaná v sekci 2.6 existují různé techniky vykreslování dat. Tyto techniky můžeme dělit dle způsobu zpracování dat na takzvané techniky Image-order a Object-order (viz Obrázek 2.10) [1]. Vykreslovací techniky jsou typicky používány k výpočtům diskrétní aproximace integrálu pro vykreslení volumetrických dat.

Dle [21], přístup Image-order pracuje ve dvojrozměrném prostoru zobrazovací plochy, kdy data zpracováváme po jednotlivých pixelech. Naproti tomu Object-order přístup zpracovává data ve 3D prostoru. Zpracovaná objemová data jsou poté promítána do zobrazovací plochy. Základní klasifikace vykreslovacích technik vypadá následovně:



Obrázek 2.10: Vykreslení dat pomocí Image-order (vlevo) a Object-order techniky (vpravo). Zdroj [1].

2.10.1 Ray casting

Ray casting, neboli sledování paprsku, již bylo popsáno na začátku kapitoly v sekci 2.4, nicméně je zde tato technika zmíněna, aby nebyla opomenuta.

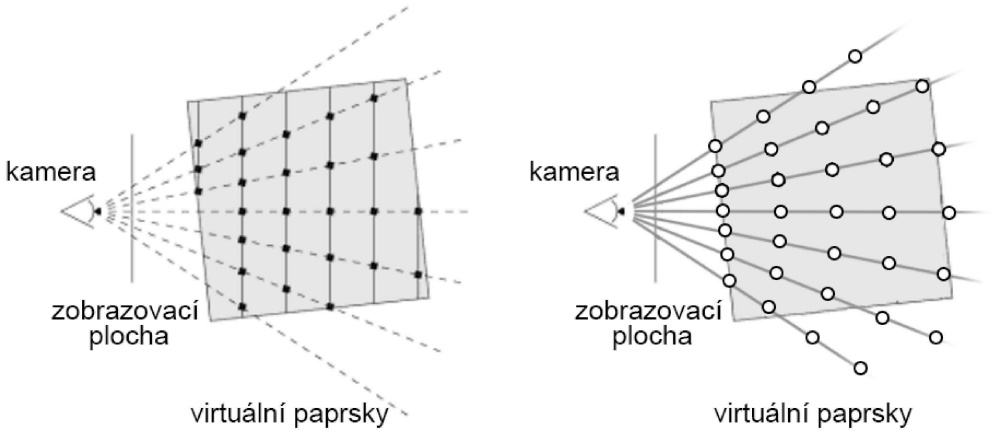
2.10.2 Texture slicing

Dle [1] se jedná o velice rozšířenou techniku uplatňovanou převážně pro vykreslování na grafických procesorech. Tento přístup generuje výsledný obraz s použitím množiny obrázků - textur. Tyto 2D řezy nacházející se ve 3D scéně vzorkují náš objekt a následně jsou na základě použitého kompozičního schématu promítnuty do zobrazovací plochy (viz Obrázek 2.11). Data jsou uložena ve 3D textuře, která reprezentuje objem objektu. Metoda vykreslení Texture slicing spadá pod Object-order přístup.

Výhodou je široká podpora ze strany hardwaru díky nenáročným požadavkům (vyžaduje pouze podporu texture a blending) [54]. Díky tomuto faktu byla donedávna technika Texture slicing dominantní metodou ve vykreslovaní volumetrických dat na GPU. Častý problém, se kterým se můžeme setkat, je vznik artefaktů pod úhlem 45 stupňů.

2.10.3 Shear-warp volume rendering

Podobně jako v předchozí technice jsou v této Object-order metodě objemová data procházena po jednotlivých řezech. Myšlenka této techniky spočívá v poloze paprsků, které



Obrázek 2.11: Princip metod texture slicing (vlevo) a ray casting (vpravo); Nalevo vidíme pohledově zarovnané řezy, vedoucí k získání vzorkovacích pozic znázorněných tečkami. Paprsky indikují, na který pixel jsou vzorky promítány. Zdroj [54]. Napravo je pro každý pixel ve scéně vyslán paprsek, podél něhož je objem tělesa vzorkován v určitých vzdálenostech.

musí být vzájemně rovnoběžné a zároveň kolmé na 2D řezy našeho objektu. Toho docílíme zkosením neboli střihem (angl. shear) daného objektu. Silnou stránkou Shear-warp volume rendering techniky je široká škála optimalizačních metod, díky čemuž je jednou z nejrychlejších metod vykreslování dat na CPU. Více o této technice lze vyčíst v publikaci [31].

2.10.4 Splatting

V metodě Splatting promítáme 3D objekty nazvané jádra (angl. kernel) na zobrazovací plochu. Těmto 2D projekcím se říká stopy (angl. footprint).

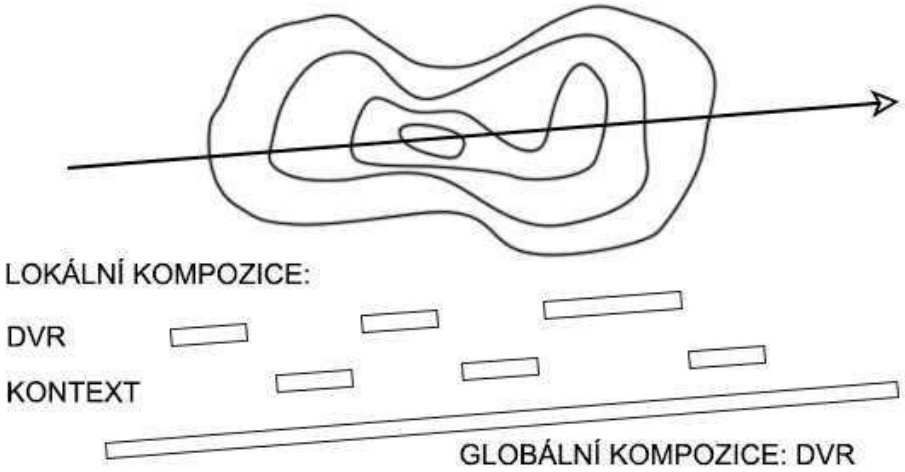
Object-order metoda Splatting byla vyvinuta za účelem zrychlení výpočtů vykreslení dat na úkor jiných atributů, například méně přesného vykreslování. Aproximováním projekce a následným složením všech stop získáme výsledný obraz. Výhodou této techniky je její malá časová náročnost, což se ovšem projeví například v přiblížených úsecích, kde může být obraz rozmazaný. Zdroj [21].

2.10.5 Cell projection

Object-order technika využívaná pro tetrahedrické a více komplexní (nerovnoměrné) mřížky. Známý je například PT algoritmus [Shirley and Tuchman].

2.11 Dvojúrovňové vykreslování dat

V této sekci se nabízí přiblížit co nového přináší segmentovaná data. Na rozdíl od mé bakalářské práce, jejímž účelem bylo rozšíření funkcionality knihovny Tiger a následná analýza a implementace metod přímého vykreslování volumetrických dat, nám nyní segmentovaná data umožňují takzvané *dvojúrovňové vykreslování*. Podstatou takového přístupu je rozdělení vykreslování do více úrovní. Myšlenka tohoto rozdělení na dvě konceptuální úrovně pro kompozici segmentovaných volumetrických dat byla poprvé popsána v [22], [20]. Obrázek 2.12 znázorňuje průchod paprsku při použití dvojúrovňového vykreslování u objektu segmentovaného do více vrstev. První z úrovní, takzvaná lokální kompozice, skládá vzorky příslušící úseku jedné vrstvy objektu. Na Obrázku 2.12 to je provedeno metodami DVR - přímé vykreslení a IA - vykreslení na základě důležitosti (z angl. importance-aware compositing) Na tyto jednotlivé úseky skládání pomocí příslušných metod lze nahlížet jako na vzorky vyšší úrovně. Zde definují autoři článku [22] úroveň globální kompozice, která takové vzorky skládá. To je zde provedeno metodou DVR - tedy přímého vykreslení. Potenciál, který také zmiňuji v sekci 2.15 je, že globální úroveň skládání nemusí být úrovní nejvyšší. Výsledná barva a neprůhlednost může být dále využita v hierarchii kompozic vyšších úrovní, což umožňuje další způsoby a kombinace vykreslení dat, a tedy rozšiřuje možnosti nahlédnutí do volumetrického modelu a pochopení jeho struktur.



Obrázek 2.12: Dvojúrovňové vykreslování volumetrických dat. Příspěvky z jednotlivých vrstev objektu jsou složeny globální kompozicí.

Nejdůležitější operací ve vykreslování segmentovaných dat, a tedy i u dvojúrovňového vykreslování, je určení příslušnosti jednotlivých fragmentů objektům čili rozhodnutí, jakého *object id* fragment nabývá, jenž se ve výsledku nemusí shodovat s *object id* voxelu (díky filtrování) [21]. Pro vykreslování tedy předpokládáme kromě obvyklých vstupů jako je hustota či hodnota gradientu také informaci o příslušnosti voxelů objektům. Tato informace je nejčastěji obsažena ve formě *segmentační masky*.

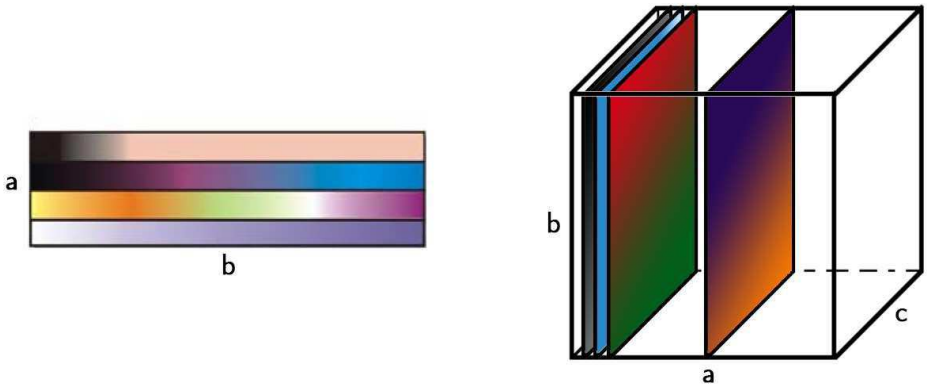
Existují dva hlavní přístupy reprezentace informace o segmentaci v maskách [21]. Za prvé může být každý objekt zastoupen binární segmentační maskou rozhodující o tom, zda-li voxel do příslušného objektu patří či nikoliv. V druhém případě, využijeme tzv. *object ID volume*, což je objemová struktura nesoucí object id všech objektů pro každý voxel. Tento přístup se dobře aplikuje na vykreslování volumetrických dat na GPU, jelikož využíváme pouze jedné objemové struktury a navíc jednotlivá ID lze jednoduše uložit do 8-bitové textury. Tato object id textura je pak následně využita k vykreslení pouze některých objektů či použití různých technik, schémat či transfer funkcí pro různé objekty. Objemová data obsahující tuto informaci jsou pak často nazývána jako "označená", lépe anglicky *tagged volumes*.

2.11.1 Operace ve fragment programu

Většina práce při vykreslování na grafické kartě se odehrává ve fragment programu (shaderu) [21]. Více o fragment shaderech bude vysvětleno v sekci 2.14.3. Nyní se pouze podívejme na operace, které jsou využívány zejména při vykreslování segmentovaných dat. Dvě základní operace jsou zamítání fragmentů, při níž nejsou fragmenty vykresleny, pokud jim nepřísluší korektní ID, a aplikování transfer funkce na úrovni fragmentů. Druhá operace je zajímavá, protože se snažíme aplikovat různé transfer funkce na více objektů v jediném vykreslovacím průchodu. Transfer funkce musí být aplikovány na individuální fragmenty na základě hodnoty dat (hustoty) a zároveň object id fragmentu. Jednou možností je tedy použít více textur pro každou z transfer funkcí. Mnohem sofistikovanější metodou je poté uchovávat transfer funkce v textuře o jednu dimenzi vyšší. Pokud tedy pracujeme s 1D transfer funkcemi, uložíme je na jedinou 2D texturu jako to vidíme znázorněno na Obrázku 2.13 (vlevo). Ve fragment shaderu je tato textura vzorkována s využitím dvojice (hustota, objectID) jako 2D souřadnice. Když je tato kombinovaná textura s transfer funkcí vzorkována, zvláštní pozornost musí být věnována správnému přiřazení transfer funkcí objektům. Vlivem lineárního interpolování se může stát, že některé transfer funkce budou aplikovány na objekty přilehlých ID. U transfer funkcí je lineární interpolování žádoucí, to ovšem neplatí pro osu určující object id, kde by se nejlépe aplikovala interpolace nejbližším sousedem. Pochopitelně není možné mít na GPU rozdílné módy pro jednotlivé osy textury. Existují ovšem dvě jednoduchá řešení, kterými lze obejít tento problém. Zaprvé lze fragment shader ponechat v původním stavu (až na indexování) a každou transfer funkci uložit v textuře dvakrát vedle sebe. Lineární interpolace dvou stejných hodnot je pak stejná hodnota a tudíž je problém vyřešen. Druhý přístup pracuje na bázi indexování přesného středu texelu, kde je pak potřeba znát hodnotu velikosti textury.

2.12 Vykreslování volumetrických dat řízené důležitostí

Dle článku Viola et al. [52] se zvyšujícího významu v posledních letech zaznamenalo v odvětví medicíny vykreslování volumetrických dat. V budoucnosti bude trojrozměrná vizualizace nezbytnou součástí lékařských diagnóz. Díky rapidnímu vývoji vysoce přesných zobrazovacích postupů se množství vyobrazených dat zvyšuje. Tudíž potřeba zdůraznit důležité části dat stoupá. V častých případech, u vyobrazování medicínských dat, se jedná o situace, kde velikost struktury, která je objektem zájmu, je poměrně malá v porovnání s celkovými daty. Příkladem mohou být nálezy nádorů v ledvinách či léze na játrech. K tomu



Obrázek 2.13: Místo užití více n-rozměrných transfer funkcí pro každý objekt, využijeme jednu texturu o n+1 rozměrech: (vlevo) jednodimenzionální transfer funkce vzorkujeme pomocí texturovacích souřadnic (*hustota, object id*), ke kterým odpovídají ve stejném pořadí osy a, b. Zdroj [20]; (vpravo) dvojdimenzionální transfer funkce vzorkujeme pomocí texturovacích souřadnic (*object id, hustota, gradient*), ke kterým v pořadí odpovídají osy a, b, c. Pro přehlednost jsou znázorněny pouze některé řezy objemem textury.

navíc nesmíme opomenout fakt, že samotná diagnostika snímků je komplexní úkol. V potaz jsou brány vlastnosti a znaky tkání, a to nejen velikost či tvar patologie, ale také jejich poloha a okolí (s ohledem na ostatní anatomické struktury). Z pohledu počítačových věd se tedy potýkáme s úkolem správného zaměření a zachování kontextu.

Přímé vykreslování je velmi silný nástroj pro pochopení volumetrických dat [52]. Většina optických modelů 2.5.2 zakomponovaných v přímém vykreslování je fyzikálně založena a považována jako 3D objekty tvořené látkou připomínající plyn, jehož částice vyzařují či pohlcují světlo. Tyto částice mají optické vlastnosti jako jsou barva či průhlednost, jejichž hodnoty jsou určovány transfer funkcemi 2.9. Realismus ovšem nemusí být nejvhodnější metodou pro jasné zobrazení informace uvnitř datového souboru. Ilustrativní (nefotorealistické) vykreslování dat nahrazuje realismus přehledností a v podstatě se skládá ze zdůrazňování důležitých vlastností a často potlačování druhotných struktur. K tomu využívá zjednodušené nebo řídké (angl. sparse) reprezentace. Mnoho ilustračních technik bylo adaptováno pro objemové vykreslování od průkopníků Rheingans – Ebert [43], a Treavett – Chen [48]. Například *Silhouette enhancement, tone and cartoon shading, halos, depth cueing, line and point drawing* jsou velmi známé techniky tohoto odvětví.

Jedna z hlavních výzev pro kvalitně informativní vykreslování je viditelnost [52]. Některé techniky ilustrativního vykreslování ovlivňují viditelnost pomocí deformací či pohledově závislých řezů k odhalení zajímavých částí objektu. Viola – Gröller [51] předložili podrobnou studii o takových přístupech ve svém článku. K zajištění viditelnosti důležitých struktur objemového tělesa, Viola et al. [53] definovali *důležitost* (angl. *importance*) jako vlastnost, udávající prioritu objemových dat a dále navrhli kompoziční schéma pracující s touto priori-

tou. Jejich metoda zprůhlední ty části, které zakrývají více důležité struktury. Já budu dále analyzovat podobnou metodu, představenou v článku [12], ze kterého vychází následující text.

2.12.1 Kompoziční schéma

Přímé vykreslování je obvykle založeno na front-to-back metodě, či back-to-front metodě skládání dat 2.6. Front-to-back schéma je zřejmým kandidátem u vykreslování objemových dat. Pozměníme-li notaci z Rovnic 2.17 dostáváme:

$$\begin{aligned} C_{i+1} &= C_i + (1 - \alpha_i)C_s, \\ \alpha_{i+1} &= \alpha_i(1 - \alpha_s) + \alpha_s. \end{aligned} \tag{2.21}$$

Zde C odpovídá barvě a α neprůhlednosti. Index i a $i + 1$ značí akumulovanou hodnotu při zpracování i-tého vzorku a akumulovanou hodnotu při zpracování vzorku na pozici $i + 1$, index s pak označuje hodnotu vzorku.

Kompoziční schéma pracující s významem jednotlivých částí musí zajistit, aby méně důležité regiony nezakrývaly ty s vyšší prioritou. Naším cílem je tedy představit front-to-back kompoziční schéma, které bude s jednotlivými vzorky a jejich viditelností nakládat na základě právě této priority.

2.12.1.1 Řízení viditelnosti

V následujícím textu je viditelnost brána jako hodnota na intervalu $[0, 1]$, která udává množství zakrytí vzorku. Jeho hodnota je rovna jedna mínus akumulovaná průhlednost ze všech vzorků spočítaných podél paprsku. Viditelnost vzorku s indexem $i + 1$ je tedy $1 - \alpha_i$. Tím pádem můžeme měnit viditelnost dalšího vzorku modulováním akumulované průhlednosti α_i. Jakákoliv změna aplikovaná na α_i také ovlivní barvu C_i. Tato změna počítá všechny předešlé vzorky jako vzorek jeden s hodnotou barvy C_i a neprůhledností α_i - akumulované hodnoty.

Naším cílem je řídit viditelnost každého příchozího vzorku na základě *důležitosti* a akumulované hodnoty důležitosti předchozích vzorků. V článku [12] tedy autoři zavádí funkci $vis(I_s, I_i)$, kde I_s je aktuální hodnota důležitosti vzorku a I_i je akumulovaná hodnota důležitosti vzorků. Tato funkce vytváří minimální viditelnost potřebnou pro daný vzorek. Pokud vyžadovaná viditelnost nemůže být uskutečněna (kvůli již naakumulované neprůhlednosti), musíme tuto neprůhlednost a barvu utlumit k zajištění správné viditelnosti vzorku. Ovšem pro smysluplné skládání kompozičního schématu na základě důležitosti nesmí být provedena žádná změna akumulovaných hodnot, je-li příchozí vzorek méně významný, než-li hodnota akumulované důležitosti. Tohoto utlumení je docíleno pomocí faktoru m, definovaného v Rovnici 2.22, jež je aplikována na akumulovanou hodnotu barvy a neprůhlednosti v každém

kroku kompozice. Faktor m_i (spočítaný po i-tém kompozičním kroku) závisí na akumulované neprůhlednosti (α_i), důležitosti vzorku (I_s) a akumulované důležitosti (I_i).

$$m_i = \begin{cases} 1 & \text{pokud } I_s \leq I_i, \\ 1 & \text{pokud } 1 - \alpha_i \geq vis(I_s, I_i), \\ \frac{1-vis(I_s,I_i)}{\alpha_i} & \text{jinak.} \end{cases} \tag{2.22}$$

Jelikož m je modulačním faktorem (angl. scale/modulation factor), Rovnice 2.22 udává, že akumulovaná hodnota barvy a neprůhlednosti bude změněna pouze, pokud důležitost příchozího vzorku je větší, než-li dosud naakumulovaná hodnota důležitosti a viditelnost umožněná nashromážděnou neprůhledností je menší, než-li vyžadovaná viditelnost. Zavedením faktoru m do původního front-to-back kompozičního schématu 2.21, kde počítáme s důležitostí jednotlivých vzorků, získáme kompoziční schéma řízené důležitostí 2.23, jež je platné pro neprůhledné vzorky.

$$\begin{aligned} C_{i+1} &= mC_i + (1 - m\alpha_i)C_s, \\ \alpha_{i+1} &= m\alpha_i(1 - \alpha_s) + \alpha_s, \\ I_{i+1} &= max(I_i, I_s). \end{aligned} \tag{2.23}$$

2.12.2 Aktualizace akumulované důležitosti

V této podsekci si přiblížíme zobecnění rovnice pro aktualizaci akumulované důležitosti, jež udělali na základě analýzy[4] autoři v článku [12]. Připomeňme, že za funkci viditelnosti $vis(I_s, I_i)$ se volí exponenciální křivka definovaná jako $vis(I_s, I_i) = 1 - exp(I_s - I_i)$, která vychází z analýzy skládání plně neprůhledných vzorků.

Požadované chování (při situaci kdy skládáme vzorky se stejnou důležitostí poté, co byly akumulovány vzorky o vysoké neprůhlednosti a nižší důležitosti) je, aby modulace, následovaná složením prvního vzorku S_J, vyprodukovala přesnou viditelnost vyžadovanou dalším, stejně důležitým vzorkem S_K. Takový požadavek reflektuje Rovnice 2.24 (první).

$$\begin{aligned} 1 - \alpha_{i+1} &= vis(I_K, I_{i+1}), \\ \alpha_J + (1 - \alpha_J)exp(I_i - I_J) &= exp(I_{i+1} - I_K). \end{aligned} \tag{2.24}$$

Pomocí Rovnic 2.22, 2.23 (druhá) převedeme první rovnici na druhou. Vezmeme-li v potaz, že hodnoty důležitosti vzorků jsou stejné ($I_J = I_K$), můžeme tuto rovnici vyřešit pro $I_i + 1$. To vede na Rovnici 2.25, kde se aktualizuje akumulovaná hodnota důležitosti na základě jejího předchozího stavu (I_i) a aktuální důležitosti a neprůhlednosti vzorku (I_s a α_s). Maximum ve funkci zaručuje, že hodnota akumulované důležitosti nikdy neklesne. Je

[4] v této analýze autoři skládali vzorky se stejnou důležitostí poté, co byly akumulovány vzorky o vysoké neprůhlednosti a nižší důležitosti

nutno poznamenat, že naprosto průhledný vzorek nijak neovlivní akumulovanou důležitost a u plně neprůhledného vzorku je akumulovaná důležitost rovna důležitosti vzorku.

$$I_{i+1} = max(I_i, ln(\alpha_s + (1 - \alpha_s)exp(I_i - I_s)) + I_s). \tag{2.25}$$

Toto vede na kompletní kompoziční schéma řízené důležitostí formalizované níže 2.26.

$$\begin{aligned} C'_{i+1} &= mC_i + (1 - m\alpha_i)C_s, \\ \alpha'_{i+1} &= m\alpha_i(1 - \alpha_s) + \alpha_s, \\ \alpha_{i+1} &= \alpha_i(1 - \alpha_s) + \alpha_s, \\ C_{i+1} &= \begin{cases} 0 & \text{pokud } \alpha'_{i+1} = 0, \\ \frac{\alpha_{i+1}C'_{i+1}}{\alpha'_{i+1}} & \text{jinak.} \end{cases} \\ I_{i+1} &= max(I_i, ln(\alpha_s + (1 - \alpha_s)exp(I_i - I_s)) + I_s). \end{aligned} \tag{2.26}$$

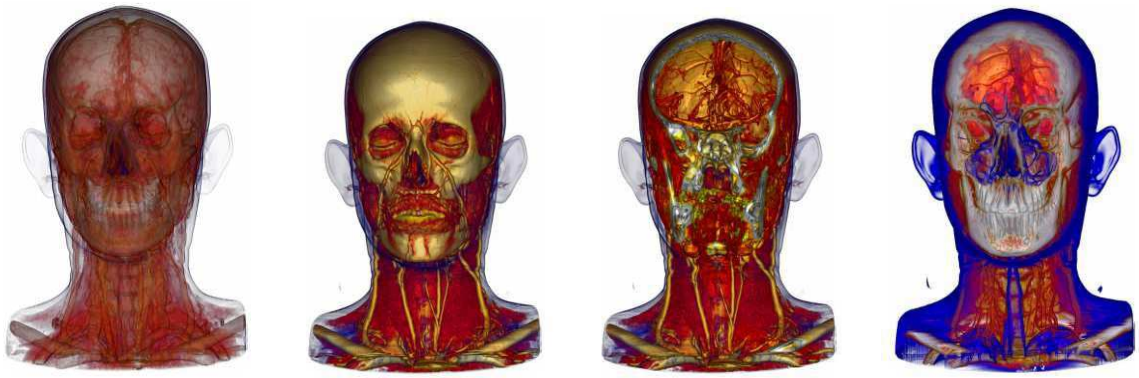
Zde je neprůhlednost α_{i+1} vypočítána z klasického skládání vzorků u front-to-back metody. Její hodnota je následně použita k modifikaci akumulované hodnoty barvy C_{i+1}. C_s a α_s jsou vlastnosti aktuálního vzorku a C'_{i+1} a α'_{i+1} jsou pomocné proměnné, které řeší problém skládání vzorků s nízkou hodnotou neprůhlednosti a vysokou hodnotou důležitosti následované vzorkem o vysoké neprůhlednosti a téměř zanedbatelné hodnotě důležitosti. To je podrobněji popsáno v článku [12].

2.13 Ilustrativní vykreslování dat zachovávající kontext

Dle Bruckner et al. [8] je ve vykreslování volumetrických dat velmi obtížné vizualizovat zároveň vnitřní a vnější struktury objektů při zachování jasných tvarů těchto struktur. Transparentní transfer funkce produkují nepřehledné obrázky s mnoha překrývajícími se strukturami. Naopak techniky, kdy se ořezávají pouze některé části mohou odstranit podstatnou informaci udávající kontext. V této sekci popíši model pro vykreslování volumetrických dat, představený autory Bruckner et al. v článku [8], který je hlavním zdrojem informací pro tuto sekci. Model pro *vykreslování zachovávající kontext* využívá funkce pro intenzitu stínování, velikost gradientu, vzdálenost od kamery a předešlou naakumulovanou neprůhlednost k utlumení neprůhlednosti v méně důležitých částech objektu. Je řízen dvěma parametry, které ovlivňují jeho chování. Tato metoda představuje alternativu ke konvenčním ořezovým technikám, obsahuje intuitivní ovládání pro uživatele a její hlavní výhodou je zachování informace o kontextu objektu.

2.13.1 Úvod do problematiky

V přímém vykreslování volumetrických dat teoreticky každý vzorek přispívá do výsledného obrázku, což umožňuje současně vykreslovat povrchy a vnitřek tělesa. V praxi je ovšem



(a) (b) (c) (d)

Obrázek 2.14: Různé přístupy ve vykreslení volumetrického souboru dat lidské hlavy; Modulace neprůhlednosti na základě velikosti gradientu (a), Přímé vykreslování volumetrických dat (b), Přímé vykreslování volumetrických dat s ořezovou rovinou (c), vykreslování volumetrických dat zachovávající kontext (d).

velmi obtížné a časově náročné přesně určit odpovídající transfer funkci. Díky exponenciálnímu růstu neprůhlednosti [25], části zakryté dalšími semi-transparentními objekty, jsou velmi obtížně rozpoznatelné. Snížení neprůhlednosti vede k úbytku příspěvku stínování u každého vzorku, což způsobuje ztrátu kontextu u tvarů, jako je to vyobrazeno na Obrázku 2.14a. Jedním z přístupů, jak obejít tento problém, je užití transfer funkcí s prudkým přechodem hodnot neprůhlednosti (angl. steep transfer functions). Tím lze dosáhnout lepšího vjemu hloubky na úkor zakrytí informace v některých regionech objemu. To můžeme pozorovat na Obrázku 2.14b.

Již zmíněnou technikou, která je často v této situaci využívána, je tím pádem ořezávání. To umožňuje odříznout vybrané části na základě pozic voxelů v datovém souboru. Problém u těchto technik je, že kromě pozice dat nemáme žádnou informaci o oříznutých částech, neboli neberou v potaz vlastnosti či rysy objemových dat[5]. V důsledku toho mohou ořezové techniky odstranit důležitou informaci o kontextu objektu vedoucí k matoucím nebo částečně zavádějícím výsledkům, které můžeme pozorovat na Obrázku 2.14c.

K vyřešení výše popsaných problémů navrhují autoři článku [8] potlačit ty regiony, které neobsahují signifikantní rysy při průchodu objemem tělesa. Hlavní myšlenka této metody je založena na pozorování, kdy velké oblasti s vysokou intenzitou osvětlení obvykle odpovídají spíše homogenním oblastem, které neobsahují takové rysy. Naopak pozice a tvar např. spekulárních odlesků dávají tělesu podnět pro vnímání zakřivení povrchu. Oblast uvnitř tohoto odlesku může být použita pro zobrazování další informace. Autoři navrhují tuto oblast udělat

[5]Tuto vlastnost lze anglicky pojmenovat data-awareness.

transparentní a umožnit tak uživateli nahlédnout dovnitř objektu. S použitím této klasifikace založené na osvětlení objektu, můžeme napodobit techniku zvanou *ghosting*. Tento přístup umožňuje ořezávání zachovávající kontext úpravou parametrů modelu. Obrázek 2.14d pak ukazuje očekávaný výsledek při použití takového modelu - cévy uvnitř hlavy jsou vyobrazeny, ačkoliv kontext dat zůstává zachován.

2.13.2 Model

Osvětlení hraje zásadní roli ve vykreslování povrchu pomocí ilustrativních metod [8]. Přístup u tohoto modelu je takový, že intenzita osvětlení slouží jako vstupní parametr funkce, která mění hodnotu neprůhlednosti na základě této informace. Neboli použijeme výsledek takové funkce ke klasifikaci jednotlivých rysů objektu.

Předpokládejme pole skalárních hodnot $f(P_i)$. Vzorek na pozici P_i je získán jako f_{p_i}. Označme gradient na pozici P_i jako $g_{P_i} = \nabla f_{P_i}$. Dále $\hat{g}_{P_i}$ je normalizovaná hodnota gradientu a $\|g_{P_i}\|$ je velikost gradientu normalizovaná do intervalu [0..1]. Zde nula odpovídá nejnižší hodnotě velikosti gradientu a jedna nejvyšší hodnotě velikosti gradientu v množině hodnot objemového souboru. Diskrétní aproximace integrálu pro vykreslení volumetrických dat využívá front-to-back kompoziční schéma k výpočtu neprůhlednosti α_i a barvy c_i v každém kroku podél paprsku:

$$\begin{aligned} \alpha_i &= \alpha_{i-1} + \alpha(P_i)(1 - \alpha_{i-1}) \\ c_i &= c_{i-1} + c(P_i)\,\alpha(P_i)(1 - \alpha_{i-1}) \end{aligned} \tag{2.27}$$

$\alpha(P_i)$ a $c(P_i)$ jsou příspěvky neprůhlednosti a barvy na pozici P_i, α_{i-1} a c_{i-1} jsou dosud naakumulované hodnoty pro neprůhlednost a barvu.

Přímé vykreslování dat využívající stínování můžeme napsat následujícím způsobem:

$$\begin{aligned} \alpha(P_i) &= \alpha_{tf}(f_{P_i}) \\ c(P_i) &= c_{tf}(f_{P_i})\,s(P_i) \end{aligned} \tag{2.28}$$

α_{tf} a c_{tf} jsou hodnoty neprůhlednosti a barvy získané z transfer funkce. $s(P_i)$ je hodnota intenzity světla na pozici vzorku získaná z osvětlovacího modelu. Pro Blinn-Phongův model používající jeden zdroj světla vypadá $s(P_i)$ následovně:

$$s(P_i) = c_d \cdot \|\hat{L} \cdot \hat{g}_{P_i}\| + c_s \cdot (\|\hat{H} \cdot \hat{g}_{P_i}\|)^{c_e} + c_a \tag{2.29}$$

c_d, c_s, c_a jsou koeficienty difúzní, spekulární a ambientní složky světla a c_e je spekulární exponent. $\hat{L}$ je normalizovaný vektor světla a $\hat{H}$ je normalizovaný *half vector*, tj. vektor půlící

úhel mezi světelným zdrojem a pozorovatelem. Modulace neprůhlednosti může být provedena následovně:

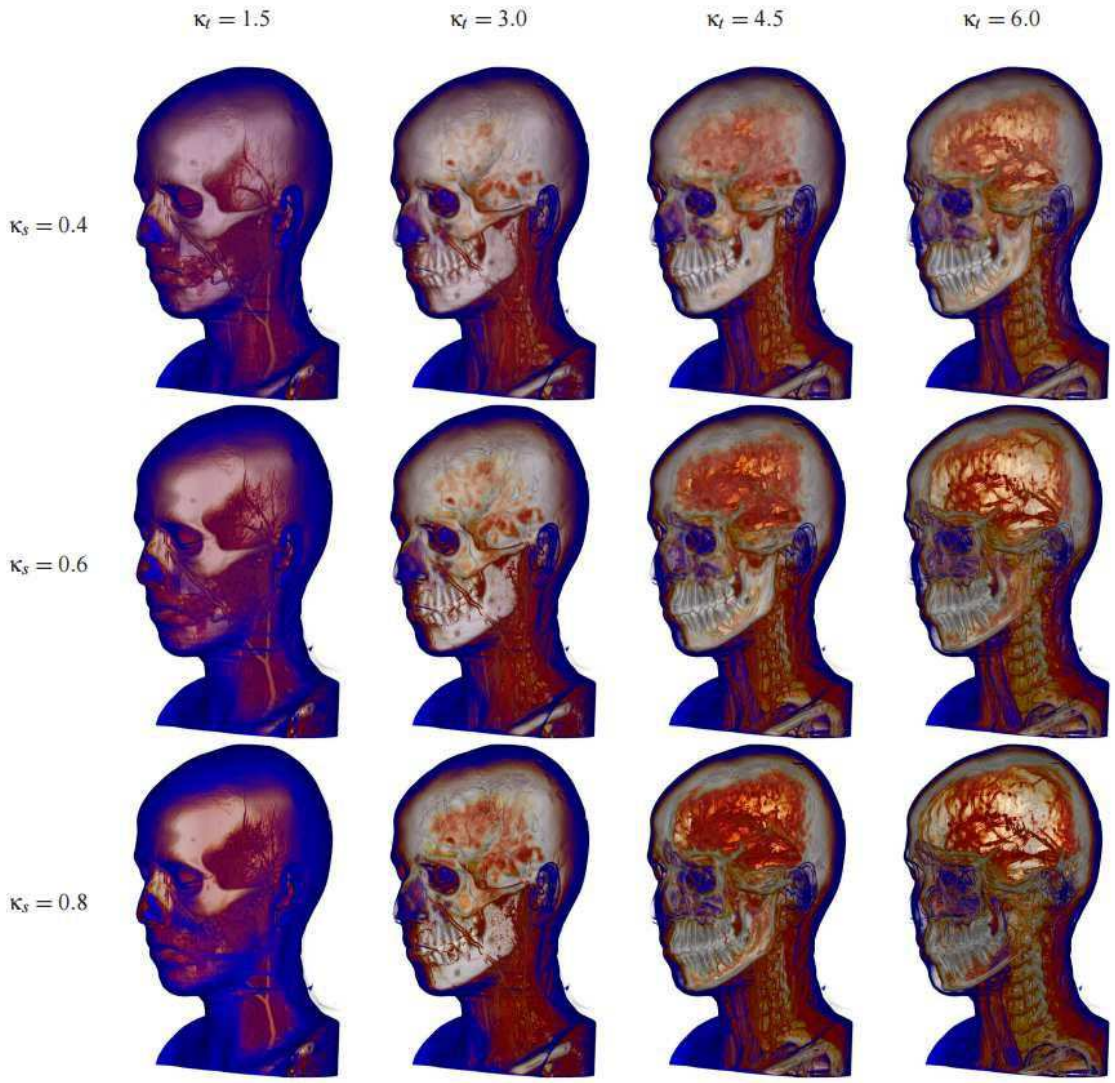
$$\alpha(P_i) = \alpha_{tf}(f_{P_i}) \cdot \|g_{P_i}\| \tag{2.30}$$

V přímém vykreslování volumetrických dat, intenzita osvětlení běžně neovlivňuje hodnotu neprůhlednosti pro daný vzorek. Velké oblasti s vyšším osvícením obvykle odpovídají spíše plochým povrchům orientovaných ke zdroji světla. Představa autorů Bruckner et al. byla taková, že tyto oblasti budou vykresleny s nižší neprůhledností. Oblasti, které jsou méně nasvícené - např. obrysy, zůstanou viditelné. Tím pádem volíme výsledek funkce $s(P_i)$ za modulační faktor neprůhlednosti. Dále chceme vzít v potaz vzdálenost od kamery a celkový efekt by také měl zohlednit velikost gradientu. Tyto požadavky nás vedou na následující rovnici pro neprůhlednost každého vzorku na pozici P_i:

$$\alpha(P_i) = \alpha_{tf}(f_{P_i}) \cdot \|g_{P_i}\|^{(\kappa_t \cdot s(P_i) \cdot (1-\|P_i - E\|) \cdot (1-\alpha_{i-1}))^{\kappa_s}} \tag{2.31}$$

Zde je $\|g_{P_i}\|$ velikost gradientu a $s(P_i)$ je intenzita stínování pro daný vzorek. Vysoká hodnota tohoto stínování indikuje místo odlesku, kde se následně sníží neprůhlednost. Výraz $\|P_i - E\|$ je vzdálenost od pozice aktuálního vzorku ke kameře normalizovaná na interval [0..1]. Efekt modelu se tedy s narůstající vzdáleností bude snižovat. Díky výrazu $1 - \alpha_{i-1}$ se struktury zakryté semi-transparentními objekty budou zdát více neprůhledné. Vliv součinu těchto tří činitelů je ovládán dvěma parametry κ_t a κ_s specifikované uživatelem. Parametr κ_t zhruba odpovídá hloubce umístění ořezové roviny (vyšší hodnoty ukazují více z vnitřku objemu). Účel druhého parametru κ_s je umožnit kontrolu nad ostrostí řezu (vysoké hodnoty budou mít za následek velmi ostré řezy, zatímco nižší hodnoty vytvářejí hladší přechody).

Obrázek 2.15 ukazuje výsledky pro různé nastavení parametrů κ_t a κ_s. Můžeme pozorovat, že s narůstající hodnotou parametru κ_t se ukazuje více z vnitřku hlavy a struktury na vnějšku hlavy jsou více a více redukovány. Zvýšení hodnoty κ_s způsobuje ostřejší přechod mezi regiony na okrajích a středu ploch natočených ke světlu.



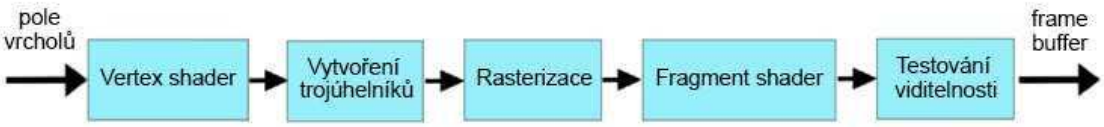
Obrázek 2.15: Ilustrativní vykreslování CT snímků hlavy zachovávající kontext při použití různých hodnot parametrů κ_t a κ_s. Snímky byly pořízeny pomocí metody vyšetření cév zobrazovací metodou zvanou angiografie. Zdroj [8].

2.14 Vykreslení geometrie na GPU

Hlavní novinkou ve funkcionalitě grafických čipů je nahrazení tradičního fixního vykreslovacího procesu programovatelným řetězcem [21]. Díky možnosti nahrání mikroprogramů na grafickou kartu, dostává programátor kontrolu nad výpočty, které probíhají na grafickém procesoru. Tyto takzvané shadery jsou na grafické kartě prováděny velmi efektivně a rychle. Programátor má dále možnost si definovat vlastní osvětlovací systém, texturování, stínování apod. Posloupnost kroků užitých k vytvoření 2D reprezentace z výchozích dat označujeme pojmem *zobrazovací řetězec.* [41]

2.14.1 Zobrazovací řetězec

Grafický procesor, který je součástí grafické karty, vykonává posloupnost příkazů, které vytvoří dvojrozměrný obraz vykreslované scény [56]. S nástupem grafických karet počítače prodělaly výraznou změnu, která uvedla v pohyb vývoj programovatelného zobrazovacího řetězce. Dalo by se říci, že se jedná o nejzásadnější změnu ve vývoji vykreslovacího řetězce. Ve fixním vykreslovacím řetězci bylo pevně dáno chování jeho jednotlivých kroků (někdy nazývaných stupňů). Programovatelný řetězec nahradil mnohé zastaralé funkce fixního řetězce, jako například okamžitého režimu (angl. immediate mode) v OpenGL [47]. Jednotlivé kroky programovatelného řetězce můžeme pozorovat na Obrázku 2.16. Nutno dodat, že proces vykreslení dat na novějších kartách je složitější, nicméně pro účely této práce bude postačovat vysvětlit metody následující:

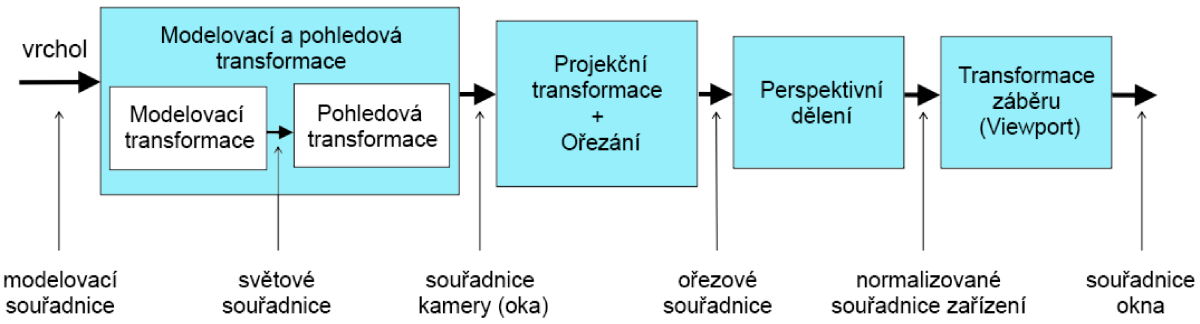


Obrázek 2.16: Programovatelný zobrazovací řetězec (angl. The programmable graphics pipeline).

Zpracování vrcholů (angl. Vertex processing)
V této části se spočítají lineární transformace příchozích vrcholů, jako například rotace, posunutí, zvětšení, případně zmenšení.

Jak můžeme vidět na obrázku 2.17, vrcholy jsou transformovány z modelovacích souřadnic do světových souřadnic (modelovací transformace), kdy umístíme a natočíme objekty ve scéně. Dále jsou vrcholy převedeny do souřadnic kamery (pohledová transformace), kdy získáme polohu a natočení kamery ve scéně a následně je aplikována projekční transformace, kde se určí typ projekce, ořezání [41]. Díky tomu se někdy zpracování vrcholů označuje obecněji jako zpracování geometrie (angl. geometry processing). Vrcholy jsou sestaveny v geometrická primitiva, na která můžeme aplikovat další funkce. Využíváme zde *vertex shaderu* 2.14.2.

Zpracování fragmentů (angl. Fragment processing)
Geometrická primitiva jsou rasterizována v takzvané fragmenty. Každý fragment připadá



Obrázek 2.17: Logické kroky při transformaci vrcholů.

jednomu pixelu výsledné plochy. Rasterizační jednotka pro každý fragment vypočítá sadu atributů, získanou interpolací příchozích atributů vrcholů. Každý fragment tedy nese sadu atributů (např. barvy, souřadnice textury atd.). Fragment shader vypočítá výslednou barvu (případně hloubku), která je poté zapsána do registrů. Zdroj [21].

Výsledná kompozice (angl. Compositing)
Zde se vyhodnocuje výsledná barva jednotlivých fragmentů pomocí *frame bufferu*, do kterého jsou poté fragmenty zapsány. Na fragmenty jsou aplikovány podmínky, které rozhodnou, zdali příchozí fragment má být vyřazen či vykreslen na obrazovce. Více se o operacích a testech spjatých s frame bufferem můžete dozvědět v sekci 2.14.4. Zdroj [21].

2.14.2 Vertex shader

Vertex shader je program, který se provede na každém vrcholu (neboli vertexu) vstupní geometrie scény. Mezi nejčastější operace ve vertex shaderu patří transformace vrcholů, které byly vysvětleny v sekci 2.14.1. Existují určitá pravidla a specifikace, které je třeba dodržovat pro správný chod vertex shaderu. Každý vstupní vrchol se musí mapovat na určitý výstupní vrchol, a zároveň tyto vrcholy mezi sebou nesdílejí informace o svém stavu. Mapování mezi vstupními a výstupními vrcholy je tedy 1:1. Jinými slovy do programu vstoupí jeden vrchol, je upraven a zase vystoupí, nelze tedy vrcholy přidávat či odebírat [6].

Příklad jednoduchého vertex shaderu vypadá následovně:

```
1  #version 330
2  in vec4 vert;
3
4  uniform mat4 projection;
5  uniform mat4 view;
6  uniform mat4 model;
7
8  void main()
9  {
10     gl_Position = projection * view * model * vert;
11 }
```

Kód 2.1: Jednoduchý vertex shader v jazyce GLSL

2.14.3 Fragment shader

Fragment shader je využíván k výpočtu výsledné barvy a v některých případech hloubky každého z fragmentů. Fragment shader je vykonán pro jednotlivé fragmenty zvlášť. Každý fragment získaný z rasterizace nese sadu atributů (např. barvy, souřadnice textury atd.). Fragment procesor pro jednotlivé fragmenty, a tedy i sady atributů, spustí fragment shader a zapíše vypočítanou barvu (případně hloubku) do registrů. Zdroj [21].

Příklad jednoduchého fragment shaderu vypadá následovně:

```
1  #version 330
2
3  void main()
4  {
5      gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
6  }
```

Kód 2.2: Jednoduchý fragment shader v jazyce GLSL

2.14.4 Frame buffer

Frame buffer je kolekce bufferů, které mohou být použity jako výstup pro vykreslování. OpenGL má dva druhy frame bufferů - defaultní, ke kterému přistupujeme pomocí OpenGL Context a uživatelem vytvořené Framebuffer objekty - ty odkazují převážně na textury [16].

Dle Hadwiger et al. [21] jsou do frame bufferu zapsány také fragmenty získané rasterizací. Ty jsou uloženy jako 2D pole atributů budoucích pixelů (barva, hloubka, neprůhlednost). Při zapsání dalšího z fragmentů upravíme dosavadní hodnoty ve frame bufferu.

Dále je možné spouštět testy, které rozhodují o zapsání fragmentu do frame bufferu. Krátký přehled vypadá následovně:

Alpha Test
Tento test zamítne fragmenty na základě porovnání hodnot průhlednosti fragmentu s určitou referenční hodnotou. Alpha Test se dá využít v mnoha směrech, avšak původním účelem této

operace bylo vyřazení zcela průhledných fragmentů.

Test šablony (angl. Stencil)
Šablona nám umožňuje označit pixely, na které aplikujeme masku uloženou v takzvaném *stencil bufferu*. Pomocí této masky se zvolí bitová rovina. U této operace používáme víceprůchodové algoritmy pro speciální efekty (obtisky, obrysy, odrazy atd.) [41].

Test hloubky (angl. Depth Test)
Jelikož geometrická primitiva nemají pevně danou sekvenci, ve které se budou vykreslovat, je třeba hloubkového testu, který zaručí správné vykreslení částečně zakrytých objektů ve scéně. Hloubka fragmentu je tedy uložena v *depth bufferu*. Tato operace testuje příchozí hloubku fragmentu a porovnává ji s hodnotou již uloženou. Zakryté fragmenty mohou být ihned vyřazeny. Tento test se často označuje jako *z-test*, jelikož rozhodnutí o vyřazení fragmentu z frame bufferu udává z-ová souřadnice.

2.15 Návrh řešení

V této sekci si popíšeme návrh řešení bez zatížení implementací. Veškeré závěry učiníme na základě analýzy.

Algoritmus vykreslení dat bude využívat metodu sledování paprsku (viz 2.4), která přímo vypočítává hodnoty podél všech paprsků. Pro každý pixel ve scéně je určen právě jeden paprsek a následně vzorkujeme v určitých intervalech objemová data našeho objektu. Program zajišťující vykreslení bude psán v jazyce Java, kde budu využívat konstruktů knihovny Tiger. Na nižší úrovni pak budeme moci nalézt příkazy OpenGL, které budou volány pomocí vazeb JOGL, a tedy i knihovnou Tiger. Implementace samotných metod bude provedena na GPU - každé metodě vykreslení bude náležet fragment shader (nebo jeho část) psaný v jazyce GLSL.

2.15.1 Struktura projektu

K tomuto projektu, jehož záměrem je vytvoření aplikace pro vykreslování volumetrických dat postavené na knihovně Tiger, přispívá více pracovníků akademické půdy. Tudíž je třeba určit a vymezit práci, která bude v rámci diplomové práce mnou provedena.

Tento dokument a s ním spjatá implementační část navazuje na mou bakalářskou práci [19]. Zde bylo úkolem rozšířit funkcionalitu stávající knihovny Tiger o práci s 3D texturami a jejich následné vyobrazení pomocí metody přímého vykreslení. Společně s tímto rozšířením jsem dále realizoval kompoziční schémata (back-to-front, front-to-back) a alternativy akumulace paprsku MIP, AIP a zapracoval vykreslování dat pomocí 1D a 2D transfer funkcí. 2D transfer funkci nebylo možné za běhu interaktivně upravovat. Následně bylo do projektu zapracováno vyobrazování popisků (angl. labeling) Bc. Martinem Jandou a interakce s daty

(změna a editace transfer funkce) v reálném čase Bc. Petrem Frantálem. V rámci letní brigády podpořené stipendiem jsem umožnil načítání formátu DICOM, což je standard pro zobrazování, distribuci, skladování a tisk medicínských dat pořízených snímacími metodami jako jsou CT, MRI či ultrazvuk [3]. Společně s tímto formátem jsem implementoval možnost načtení masky, která nese informaci o klasifikaci dat (např. rozmístění nádorů v kostech pacienta).

V rámci diplomové práce umožním segmentaci dat, jejíž účelem je přiřazení unikátních hodnot (*object id*) určitým částem objektu (viz. 2.2). Dále realizuji skládání hodnot vhodným kompozičním schématem, kde bude možné každému úseku (vymezenému stejným object id) přiřadit vlastní metodu vykreslení. K tomu implementuji a využiji dvě metody prezentované v článcích de Moura Pinto – Freitas [12], Bruckner et al. [8].

2.15.2 Postup

Stejně jako v [19] po nahrání dat na grafickou kartu začneme tím, že vykreslíme přivrácené (přední) plošky kvádru dat, díky čemuž získáme texturovací souřadnice začátku paprsku. Analogicky dostaneme texturovací souřadnice konce paprsku vykreslením odvrácených plošek. Ve fragment shaderech si poté odečtením těchto souřadnic vypočítáme směr paprsku. Budeme tedy paralelně trasovat pro každý fragment paprsek, podél něhož budeme vypočítávat výslednou barvu. Způsoby, jak vypočítat barvu výsledného pixelu, se liší podle užitého kompozičního schématu (podrobně popsáno v [19]).

Segmentace
Segmentace dat bude umožněna dvěma způsoby. První způsob je, že informace o příslušnosti objektů bude dodána ve formě segmentační masky s již nasnímanými daty. V odpovídajícím fragment shaderu pak tuto masku načteme ve formě textury. Na základě hodnot této textury pak budeme používat metody a kompoziční schémata. Jelikož jsou data nesoucí informaci o segmentaci velmi vzácná, je třeba navrhnout metodu, která vytvoří segmentační profil pokud informace o příslušnosti objektů nebude k dispozici. Základní přístup je přiřadit object id na základě skalárních hodnot dat. Realizovaná metoda potřebuje jediný parametr - počet vrstev (neboli počet úseků), na který chceme objekt rozdělit. Tento parametr budeme moci za běhu měnit pomocí GUI. Úseky hodnot object id tak půjdou za běhu upravovat.

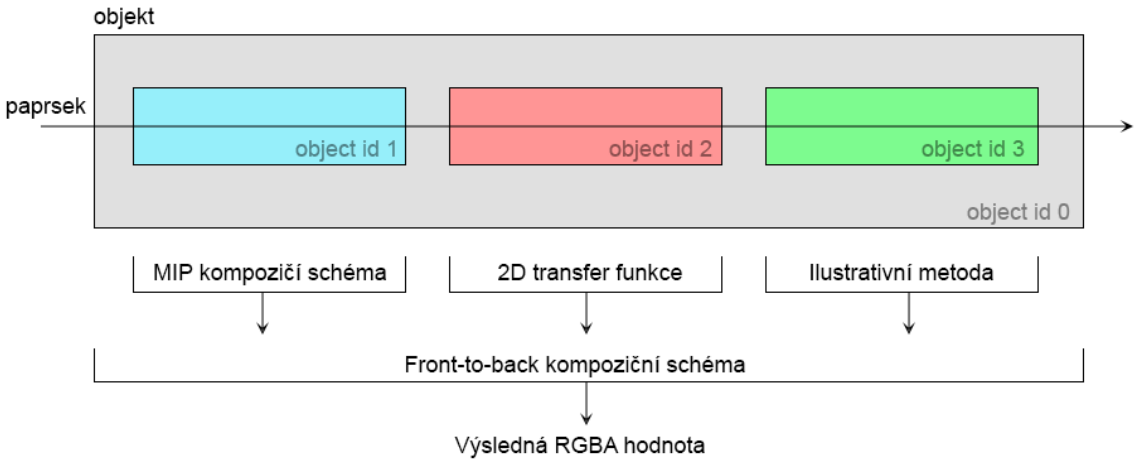
Jelikož počet metod, které lze přiřadit jednotlivým úsekům objemu není pevně daný, musíme realizovat uchovávání informace o přiřazení metod daným object id vhodným způsobem. Flexibilní a zároveň často používaná metoda při programování na GPU je uložení této informace do 2D textury, ke které pak přistupujeme uvnitř shaderu. Pseudokód k výše popsanému chování můžeme pozorovat v Algoritmu 1.

Kompozice
Aby šlo jednotlivé úseky vymezené stejnými hodnotami object id skládat dohromady, bude zapotřebí je nejprve detekovat a následně složit pomocí kompozičního schématu. Na Obrázku 2.18 můžeme pozorovat princip skládání takových úseků. Povšimněte si, že nám vzniká hierarchie kompozičních schémat. Na nejnižší úrovni skládáme jednotlivé vzorky objemu kom-

Algoritmus 1 Vykreslování segmentovaných dat

```
1: procedure SegmentovanaData
2: Pro každý vzorek podél paprsku pracuj:
3:     Načti hodnotu dat z textury.
4:     Normalizuj hodnotu dat do intervalu [0..1].
5:     Zjisti hodnotu object id na základě hodnoty dat.
6:     Zjisti typ metody na základě object id.
7:     Použij příslušnou metodu a kompoziční schéma k výpočtu barvy a neprůhlednosti.
8: end procedure
```

pozičním schématem, který určíme na základě object id (viz. pseudokód 1). Na vyšších úrovních pak skládáme tyto úseky, jejichž výstupem je hodnota barvy a průhlednosti globálním kompozičním schématem - zde volím front-to-back metodu. Potenciál tohoto hierarchického modelu je takový, že na metodu skládání segmentovaných dat lze nahlížet jako na jakoukoliv jinou metodu. Tu pak lze uvnitř hierarchie skládat pomocí vyšší úrovně.



Obrázek 2.18: Hierarchie skládání segmentovaných dat. Jednotlivé úseky vyznačené barevně mají přiřazenou metodu vykreslování. Šedý úsek bude oříznut, a proto není v hierarchii skládán.

Návrh metody vykreslování dat řízené důležitostí

První navrhovanou metodou je vykreslování volumetrických dat řízené důležitostí. K její realizaci bude zapotřebí znát hodnoty důležitosti vzorků. Tyto hodnoty mohou být dodány s daty ve formě bufferu, které následně načteme do textury a předáme fragment shaderu. Další variantou je získat důležitost z dat samotných, kde je důležitost určena ze skalárních hodnot, velikosti gradientu či jinou sofistikovanější metodou. Výpočet důležitosti musí proběhnout nemáme-li data předem k dispozici. Následně je objem, či úsek objemu (při vykreslování segmentovaných dat) vykreslován, dle schématu popsaného v Algoritmu 2.

Algoritmus 2 Vykreslování dat řízené důležitostí

1: **procedure** VykreslovaniSDulezitosti
2: Pro každý vzorek podél paprsku pracuj:
3: Načti hodnotu dat barvy a neprůhlednosti z textury.
4: Načti hodnotu důležitosti vzorku z textury.
5: Spočítej modulační faktor.
6: Spočítej složku barvy ovlivněnou modulačním faktorem podle Rovnice 2.26 (1.).
7: Spočítej složku α ovlivněnou modulačním faktorem podle Rovnice 2.26 (2.).
8: Akumuluj důležitost podle Rovnice 2.26 (5.).
9: **end procedure**

Návrh metody vykreslování dat zachovávající kontext

Vykreslování dat zachovávající kontext vyžaduje dva parametry určené uživatelem. Jsou to κ_t - určující hloubku umístění ořezové roviny a κ_s - kontrola ostrosti řezu. Po předání těchto parametrů využijeme Rovnice 2.31, podle které se aktualizuje hodnota neprůhlednosti α. Oblasti, jež jsou méně nasvícené (např. obrysy), budou zobrazeny viditelněji, než-li povrchy orientované ke zdroji světla - jejich viditelnost bude utlumena. Následující Algoritmus 3 zachycuje princip metody:

Algoritmus 3 Vykreslování dat zachovávající kontext

1: **procedure** VykreslovanZachovavajiciKontext
2: Pro každý vzorek podél paprsku pracuj:
3: Načti hodnotu dat barvy a neprůhlednosti z textury.
4: Načti hodnoty parametrů κ_t a κ_s.
5: Spočítej složku barvy.
6: Spočítej složku α ovlivněnou dvěma faktory κ_t a κ_s podle Rovnice 2.31.
7: **end procedure**

Kapitola 3

Implementace

V této kapitole jsou představeny technologie, které jsem použil v implementaci. Dále zde popíši strukturu projektu, tedy z kterých částí se skládá a naopak, které části již byly implementovány v předchozím vývoji.

3.1 Použité technologie

Kromě toho, že je nezbytné definovat, pro jaké účely bude výsledná aplikace sestavena, popsat části které má obsahovat a diskutovat metody, jež využiji k její funkcionalitě, je nutné pečlivě zvolit jaké technologie použít k její konstrukci. V této části se tedy budu věnovat technologiím použitým v implementační části. Vzhledem k tomu, že v rámci diplomové práce navazuji na předešlou činnost v podobě bakalářské práce a dalších aktivit, jež se podílely na vývoji aplikace pro vykreslování volumetrických dat, jsou technologie striktně dány:

3.1.1 OpenGL (Open Graphics Library)

Základní stavební prvek této aplikace je průmyslový standard, specifikující multiplatformní rozhraní pro tvorbu aplikací počítačové grafiky, OpenGL [38]. Jeho základní funkcí je vykreslování do obrazové paměti (angl. frame buffer). Činnost OpenGL se řídí vydáváním příkazů pomocí volání funkcí. Umožňuje nám vykreslit základní primitiva, nahrávat textury na grafickou kartu a má velkou výhodu, že implementace OpenGL existují pro prakticky všechny počítačové platformy. Jako velké plus bych zmínil rozsáhlou dokumentaci, která se nachází na oficiálních stránkách.

Na OpenGL lze nahlížet jako na automat - v každém okamžiku má definován svůj vnitřní stav, jenž je určen hodnotami vnitřních proměnných [40]. Tyto proměnné lze nastavovat a ptát se na jejich hodnotu. OpenGL je vyvíjecí se API, jehož nové verze jsou pravidelně vydávány skupinou Khronos Group. Každá taková verze rozšiřuje API o podporu pro mnoho dalších vlastností a funkcí [38].

3.1.2 JOGL (Java Binding for the OpenGL API)

OpenGL, které bylo výše popsáno, je nejrozšířenější grafickou knihovnou, která je podporována napříč všemi populárními počítačovými platformami. JOGL neboli Java Binding for the OpenGL API, implementuje Java vazby pro přístup k OpenGL [9]. Poskytuje vykreslování 3D grafiky aplikacím napsaných v jazyce Java. Je součástí skupiny zabývající se open-source technologiemi, kterou iniciovala skupina Game Technology Group v Sun Microsystems. JOGL poskytuje plný přístup k OpenGL funkcím a dále využívá grafických knihoven AWT, Swing a SWT. Pro více informací doporučuji přečíst knihu Chen – Wegman [9].

3.1.3 Tiger (Toolkit for Interactive Graphics rendERing)

Tiger je knihovna v Javě pro zjednodušení vývoje víceprůchodového vykreslování v OpenGL. Tiger využívá JOGL k přístupu k OpenGL API. [24] Na jejím vývoji jsem se podílel v rámci bakalářské práce [19] a jejím autorem je vedoucí práce Ing. Ladislav Čmolík, Ph.D.

3.1.4 GLSL (OpenGL Shading Language)

S GLSL[1] jsme se setkali v sekci 2.14.2 a 2.14.3, kde jsme si ukázali jednoduché programy operující na grafické kartě (shadery).

OpenGL Shading Language vychází ze syntaxe C, kdy tento jazyk vznikl v rámci procesu postupné transformace fixního vykreslovacího řetězce na řetězec programovatelný (programovatelná pipeline) [39]. GLSL nabízí množinu vestavěných funkcí, které operují s geometrií, úhly, maticemi a vektory či texturami. U tohoto jazyka bych zmínil speciální direktivy, což jsou příkazy pro preprocesor, který zpracovává kód před jeho spuštěním. Dvě direktivy jsem použil v mých shaderech, a to #version a #extension. Dále můžeme použít běžných direktiv z jazyka C.

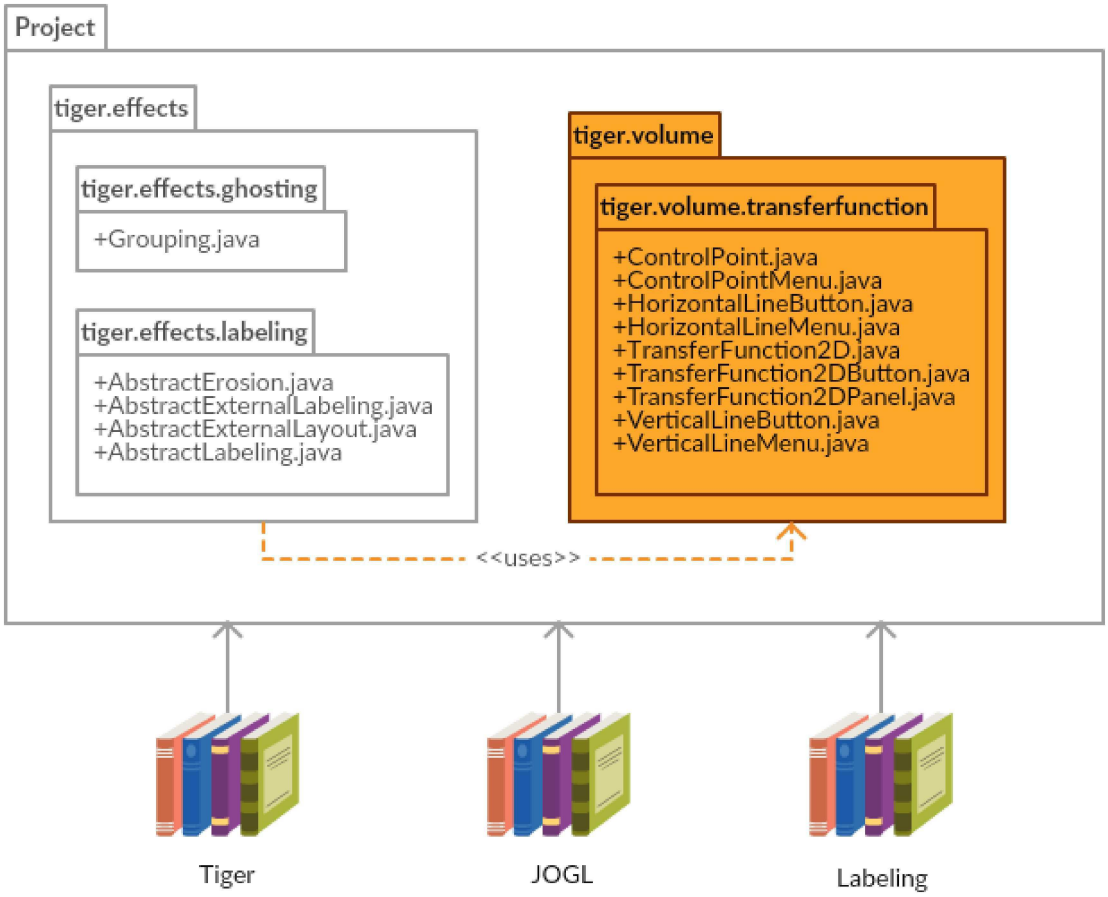
3.2 Struktura aplikace

Jak již bylo zmíněno v sekci 2.15, k tomuto projektu přispívá více pracovníků akademické půdy. Nejprve si tedy popíšeme, které balíčky aplikace jsou spjaty s mojí diplomovou prací.

Na Obrázku 3.1 jsou znázorněny balíčky v jednoduchém diagramu. Oranžově jsou označeny ty balíčky, ve kterých se soustředila má činnost. Zároveň se zde nachází externí knihovny, které využívám k běhu aplikace. Balíček `tiger.volume` obsahuje nejdůležitější třídy aplikace a jeho struktura bude popsána níže. Balíček `tiger.volume.transferfunction` zahrnuje funkcionalitu editace transfer funkce. Tato funkcionalita byla upravena, aby mohla

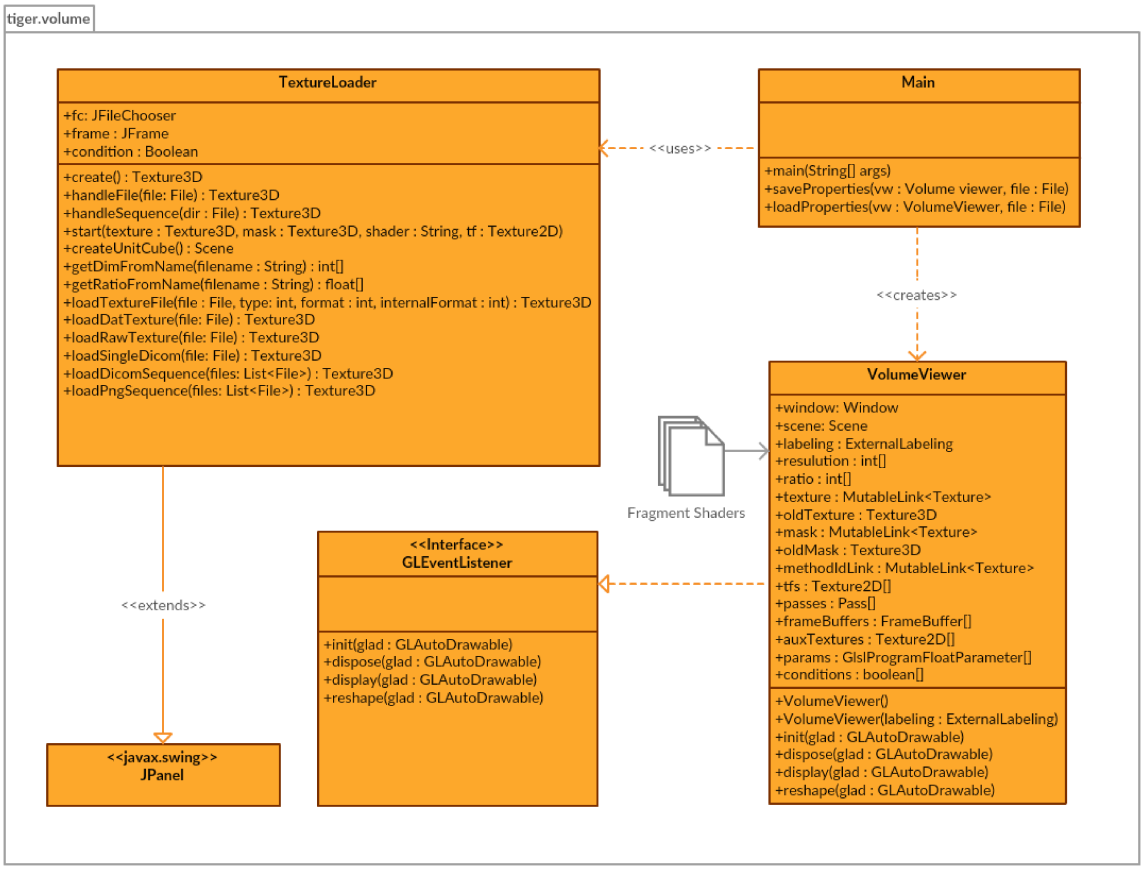
[1] Další alternativou pro GLSL je například jazyk Cg (C for Graphics), který se mírně liší v syntaxi.

být využita při mapování segmentovaných dat. Třída `TransferFunction2D`, jejíž činnost je pevně svázána s ostatními třídami tohoto balíčku, je následně použita v GUI aplikace.



Obrázek 3.1: Diagram balíčků aplikace a externích knihoven. Oranžově jsou znázorněny balíčky, kde se soustředila má činnost.

Na Obrázku 3.2 můžeme pozorovat diagram tříd balíčku `tiger.volume`. Součástí tohoto balíčku jsou také shadery, které implementují přímé a ilustrativní metody kompozice vzorků dle paprsku, které byly popsány v kapitole 2. Třída `Main` je hlavní třídou aplikace. Jejím úkolem je příprava a vytvoření GUI, pomocí kterého bude možné aplikaci ovládat. Konkrétně se jedná o vytvoření okna, obsahujícího dva oddíly - scénu a postranní lištu. V této liště se nachází panel pro editaci 2D transfer funkcí a komponenty, jako například posuvníky, pro úpravu vstupních parametrů aplikace. Stav těchto komponent je možné uložit a načíst pomocí metod `saveProperties(...)` a `loadProperties(...)`. Ty využívají javovské třídy `Properties` k uložení informací do souboru. Dalším úkolem třídy `Main` je vytvoření instance třídy `VolumeViewer`.



Obrázek 3.2: Diagram tříd balíčku `tiger.volume`. Balíček obsahuje fragment shadery, kde každý shader implementuje rozdílnou metodu vykreslení volumetrických dat.

Jak vidíme na Obrázku 3.2, `VolumeViewer` implementuje rozhraní `GLEventListener` a společně s ním metody `init(...)`, `display(...)`, `reshape(...)` a `dispose(...)`. Tyto metody zaručují správné chování okna při inicializaci, vykreslení, změně měřítka a ukončení vykreslování. Třída má dále za úkol přípravu textur, frame bufferů, parametrů a shaderů, které předá grafické kartě vždy jedním průchodem, pomocí třídy `Pass`. K nahrání textur ze souboru slouží třída `TextureLoader`. Ta je volána z hlavní třídy programu a výsledné textury jsou předány do třídy `VolumeViewer`. `TextureLoader` je třída dědící z javovské třídy `JPanel`. Funkcionalita této třídy se dělí na nahrávání textury z jednoho souboru a na nahrání ze sekvence řezů různých formátů. K tomu slouží metody `handleFile(file : File)` a `handleSequence(file : File)`. Na základě typu souboru či řezů vstupních dat je načtena textura ve formátu `Texture3D`, kterou lze předat na grafickou kartu. Možné formáty načítaných dat jsou následující: raw 8-bit, raw 16-bit, dat, Dicom; sekvence: png, Dicom.

Přehled implementovaných shaderů, které naleznete v tomto balíčku jsou vypsány v Tabulce 3.1:

Název	Implementovaná metoda
showTF.frag	Zobrazení 2D transfer funkce užité pouze pro testovací účely.
cube.frag	Jednoduchý shader užitý při vykreslení základní krychle.
dicom_shader.frag	Shader implementující přímou metodu vykreslení mapující data pomocí 1D i 2D transfer funkce.
dicom_shader_MIP.frag	Maximum Intensity Projection. Zobrazuje maximální hodnotu podél paprsku.
dicom_shader_context.frag	Ilustrativní vykreslování dat zachovávající kontext.
dicom_shader_data_data.frag	Dvojúrovňové vykreslování segmentovaných dat.
dicom_shader_importance.frag	Vykreslování volumetrických dat řízené důležitostí.
dicom_shader_mask.frag	Přímé vykreslování volumetrických dat s klasifikovanou maskou.
empty.frag	Shader použitý u aplikace bez načteného volumetrického modelu.

Tabulka 3.1: Přehled implementovaných shaderů. Soubory s předponou dicom byly primárně zaměřeny na zobrazování volumetrických dat ve formátu Dicom, následně bylo jejich použití rozšířeno na všechny podporované formáty.

3.3 Implementační detaily

3.3.1 Gradient

Stínování volumetrických dat, které jsem použil u obou přístupů - tedy přímého a ilustrativního vykreslování, vyžaduje určit pro každý vzorek podél paprsku, normálu povrchu. Tu lze získat z gradientu ve vrcholech pravidelné mřížky. Gradient odpovídá směru a velikosti největší změny skalárního pole a uvnitř shaderu je vypočítán následovně:

```
1  vec3 calculateGradient(vec3 texFront) {
2      float valueXa = texture3D(texture, texFront+vec3(+stepSize,0,0)).x;
3      float valueXb = texture3D(texture, texFront+vec3(-stepSize,0,0)).x;
4      float valueYa = texture3D(texture, texFront+vec3(0,+stepSize,0)).x;
5      float valueYb = texture3D(texture, texFront+vec3(0,-stepSize,0)).x;
6      float valueZa = texture3D(texture, texFront+vec3(0,0,+stepSize)).x;
7      float valueZb = texture3D(texture, texFront+vec3(0,0,-stepSize)).x;
8
9      return vec3((valueXb-valueXa)/2.0,(valueYb-valueYa)/2.0,
10         (valueZb-valueZa)/2.0);
11 }
```

Kód 3.1: Získání gradientu ze skalárních dat.

Zde *texFront* udává vstupní hodnotu vzorku podél paprsku, *stepSize* značí délku kroku, neboli vzdálenost mezi vzorky. *Texture* nese informace objemové textury, která byla načtena ze souboru a nahrána na grafickou kartu. Pokud vypočteme gradient pro každý vrchol pravidelné mřížky, můžeme vypočíst pro každý vzorek normálový vektor a pak změnit barvu získanou z mapování pomocí osvětlovacího modelu (např. Phongova). Tím aplikujeme stínování

a budeme schopni lépe vnímat struktury v datech. Gradient, respektive velikost gradientu, jsem využil nejen pro stínování, ale také pro mapování ve 2D transfer funkci. Použití velikosti gradientu pro mapování vypadá následovně:

```
1 float rangeInv = 1.0/(maximum-minimum);
2 vec3 normal = calculateGradient(texFront);
3 vec3 gradientMag = (length(normal)-minimum)*rangeInv;
4 vec3 color = texture2D(tf, vec2(data, gradientMag)); //transfer function
5 float sp = blinnPhong(normal, normalize(dir), color.rgb); //shading
6 color = color * vec4(sp,sp,sp,1.0);
```

Kód 3.2: Použití velikosti gradientu.

Nejprve je nutné normalizovat velikost gradientu do intervalu [0..1], čehož je docíleno proměnnými *rangeInv*, *maximum* a *minimum*, které udávají rozsah hodnot množiny dat. Dále je na řádku 4 mapována barva pomocí 2D transfer funkce na základě hodnoty skalárních dat a velikosti gradientu. Řádky 5 a 6 aplikují Blinn-Phongovo stínování na volumetrický model.

3.3.2 Stínování

Stínování těles je založeno na Phongově a Blinn-Phongově modelu. Tyto techniky stínování využívají výše popsaného gradientu, a tedy normály povrchu těles. Příchozí světlo je dáno staticky a určuje ho vektor (0.0, -1.0, 0.0). Následující dva úseky kódu implementují stínování použité v diplomové práci:

```
1  float blinnPhong(vec3 gradient, vec3 direction, vec3 color) {
2      float specular = 0.0;
3      // Normal vector
4      vec3 N = normalize(gradient);
5      // Light vector
6      vec3 L = vec3(0.0, -1.0, 0.0);
7      float lambertian = abs(dot(L, N));
8      //Half vector
9      vec3 H = vec3(1.0);
10     if(lambertian > 0.0){
11         vec3 V = direction;
12          H = normalize(L + V);
13         float specAngle = max(dot(H, N), 0.0);
14         specular = pow(specAngle, shininess);
15     }
16     return lambertian + specular;
17 }
```

Kód 3.3: Stínování dle Blinn-Phongova osvětlovacího modelu. *Shininess* je parametr udávající hodnotu lesku tělesa.

```
vec3 phong(vec3 gradient, vec3 color) {
    // Normal vector
    vec3 NN = normalize(gradient);
    // Light vector
    vec3 NL = vec3(0.0, -1.0, 0.0);
    float NdotL = abs(dot(NN, NL));
    // apply Phong shading model
    if(gradient != 0.0){
        color.rgb*=(NdotL+(1.0-phongParam-NdotL*(1.0-phongParam));
    }
    return color;
}
```

Kód 3.4: Stínování dle Phonogova osvětlovacího modelu. *PhongParam* je parametr, udávající míru s kterou bude stínování použito.

3.3.3 Přímé vykreslování

Metodu přímého vykreslování volumetrických dat jsem implementoval již pro svou bakalářskou práci. Tuto metodu využívám u dvojúrovňového vykreslování segmentovaných dat. Také lze přímé vykreslování kombinovat s ostatními ilustrativními metodami. Kód, který pomocí transfer funkce složí vzorek přímým vykreslením na základě akumulovaných hodnot barvy a neprůhlednosti, vypadá následovně:

```
vec4 transferFunction(vec4 cout, vec4 color){

    cout.rgb += (1.0 - cout.a) * color.rgb * color.a;
    cout.a += (1.0 - cout.a) * color.a;

    return cout;
}
```

Kód 3.5: Přímá metoda vykreslování volumetrických dat. Vstupní parametry funkce `transferFunction(...)` je dosud naakumulovaná barva a neprůhlednost *cout* a dále aktuální barva a neprůhlednost vzorku *color*.

3.3.4 Ilustrativní vykreslování

3.3.4.1 Vykreslování dat řízené důležitostí

Metoda vykreslování řízeného důležitostí byla implementována formou jednoho průchodu ve fragment shaderu `dicom_shader_importance.frag`.

Modulace neprůhlednosti

Základem metody je modulace již naakumulované neprůhlednosti pomocí empirické funkce, kterou autoři [53] definovali pomocí exponenciální funkce závislé na parametrech akumulované důležitosti *ii* a důležitosti aktuálního vzorku *is*. Implementace takových funkcí vypadá následovně:

```
1 float vis(float is, float ii){
2     return 1.0 - exp(ii - is);
3 }
4
5 float modulation(float opacity, float is, float ii){
6     if(is <= ii) return 1.0;
7     if(1.0 - opacity >= vis(is, ii)) return 1.0;
8     return (1.0 - vis(is, ii))/opacity;
9 }
```

Kód 3.6: Modulace akumulované neprůhlednosti na základě akumulované důležitosti *ii* a důležitosti aktuálního vzorku *is*.

Neprůhlednost *opacity* na řádku 8 může nabývat hodnoty 0.0. Jelikož bychom uvnitř fragment shaderu dělili nulou, je toto v kódu ošetřeno pomocí funkce `max()`. Tím se vyhneme nedefinovanému chování, ovšem pro přehlednost, zde uvádím kratší verzi.

Měření důležitosti dat

Celkem bylo implementováno 5 různých přístupů k měření důležitosti ze skalárních dat. První z přístupů přiřazuje vyšší hodnoty důležitosti siluetám objektu. Toho je dosaženo díky skalárnímu součinu vektorů *dot(V,N)*, což je základní kámen většiny technik zobrazující siluety. Dále je vykreslování siluet ovlivněno velikostí gradientu *gradientMag*. Do výpočtu vstupuje trojice parametrů, v kódu označené jako *p1, p2* a *p3*:

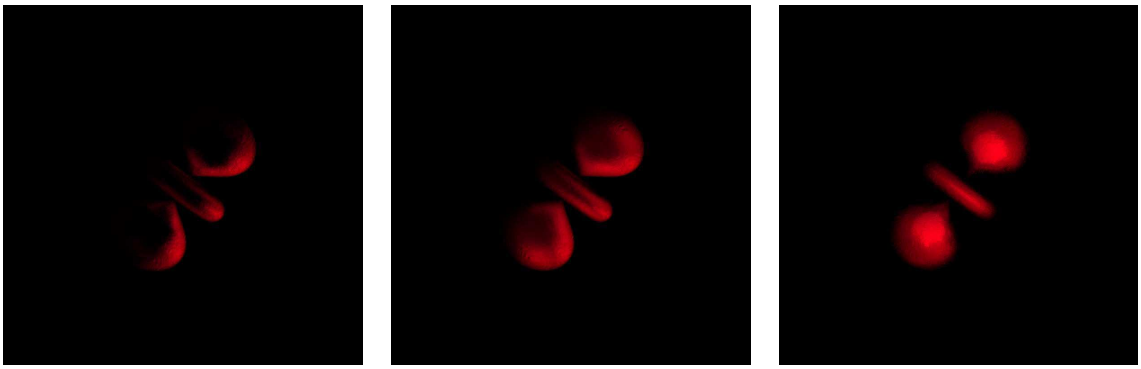
```
1 float importanceSil(vec3 gradient, vec3 direction, float gradientMag){
2
3     vec3 N = normalize(gradient);
4     vec3 L = vec3(0.0, -1.0, 0.0);
5     vec3 V = direction;
6
7     return pow(gradientMag,p1) * smoothstep(p2,p3,(1.0-abs(dot(V,N))));
8 }
```

Kód 3.7: Měření důležitosti dat na základě siluet.

Parametr *p1* ovlivňuje velikost gradientu. Parametry *p2* a *p3* vstupují do vestavěné funkce jazyka GLSL `smoothstep(a, b, x)`, která provede Hermitovu interpolaci mezi hodnotami a a b. Na Obrázku 3.3 můžeme pozorovat vliv parametrů na model atomu vodíku. Vidíme, že první z parametrů ovlivňuje, jaké části tělesa se zobrazí. Nízké hodnoty mají za následek vykreslení pouze těch částí, které obsahují vyšší hodnotu velikosti gradientu. Zbylé dva parametry modifikují ostrost přechodu hran. Při nastavení parametrů pro které platí, že $p3 > p2$ dostáváme opačný efekt, tj. útlum siluet a vykreslení mezilehlých částí.

Vykreslení siluet lze dále umocnit použitím vhodně nastavené exponenciální funkce. Výslednou barvu vzorku ovlivňuji následovně: $color.rgb* = exp(-8 * sil)$, kde *sil* odpovídá důležitosti vzorku.

Dalším přístupem měření důležitosti dat je přiřazovat důležitost na základě osvětlení. Tohoto docílíme přiřazením vyšších hodnot důležitosti vzorkům, kde je Blinn-Phongovo osvětlení utlumené. H je v následujícím kódu half vector, tj. vektor půlící úhel mezi světelným zdrojem a pozorovatelem, *directon* je směr putování paprsku od pozorovatele. Ostatní proměnné mají stejný význam jako v předchozím textu. Důležitý je empiricky získaný výraz na



(a) p1=0.1, p2=0.5, p3=0.2 (b) p1=0.4, p2=0.5, p3=0.2 (c) p1=0.1, p2=0.5, p3=0.55

Obrázek 3.3: Vliv parametrů při vykreslování na základě důležitosti změřené ze siluet dat.

řádcích 12-13 v kódu níže. Zde parametry *p1*, *p2* a *p3* ovlivňují spekulární a difúzní složku a dále velikost gradientu.

```
1  float importanceLight(vec3 gradient, vec3 direction, float gradientMag){
2
3      vec3 N = normalize(gradient);
4      vec3 L = vec3(0.0, -1.0, 0.0);
5
6      float lambertian = max(dot(L, N),0.0);
7      vec3 H = vec3(0.0);
8      if(lambertian > 0.0){
9          vec3 V = direction;
10         H = normalize(L + V);
11     }
12     return 3.0 - pow(max(dot(N,H),0.0),p1) - pow(max(dot(N,L),0.0),p2)
13     - pow((1.0 - gradientMag),p3);
14 }
```

Kód 3.8: Měření důležitosti dat na základě osvětlení.

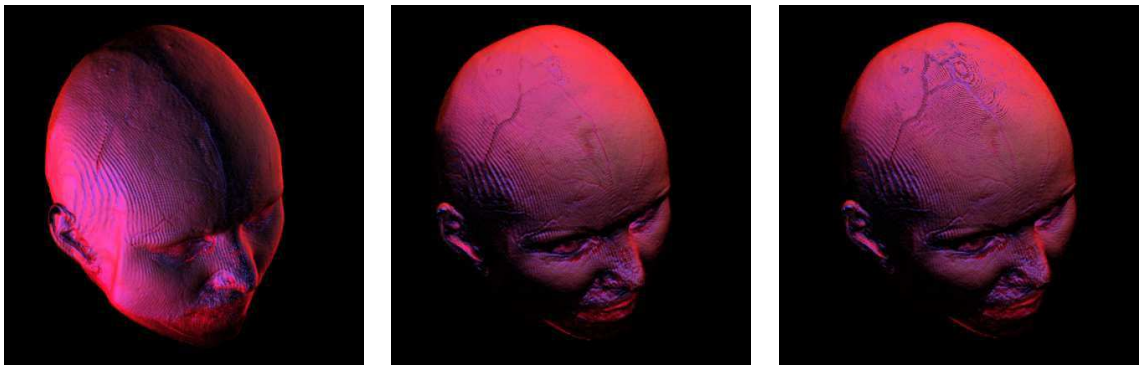
Na Obrázcích 3.4 vidíme model ženské hlavy. Změnou druhého parametru bylo docíleno přiřazení větší důležitosti vzorkům odpovídající struktuře lebky. Ty jsou znatelnější na posledním snímku 3.4c. První snímek byl vykreslen klasicky přímou metodou.

Další měření důležitosti byla provedena v závislosti na intenzitě dat, velikosti gradientu a prahového rozdělení, kdy jednotlivým úsekům intenzit (vrstvám) byla přiřazena určitá důležitost.

3.3.4.2 Vykreslování dat zachovávající kontext

Metoda vykreslování dat zachovávající kontext byla implementována formou jednoho průchodu ve fragment shaderu `dicom_shader_context.frag`.

Implementované kompoziční schéma můžeme vidět níže.



(a) DVR (b) IA, p2 = 10 (c) IA, p2=100

Obrázek 3.4: Model ženské hlavy. (a) Přímé vykreslení dat, (b) a (c) vykreslení na základě důležitosti řízené osvětlením.

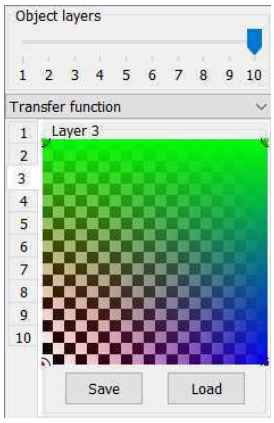
```
1 cout.rgb += (1.0 - cout.a) * color.rgb * color.a;
2 color.a *= pow(gradientMag,
3                pow(kappa_t*sp*(1-distToCamera)*(1-cout.a),kappa_s));
4 cout.a += (1.0 - cout.a) * color.a;
```

Kód 3.9: Kompoziční schéma metody vykreslování zachovávající kontext.

Proměnné zde mají význam: *cout* je akumulovaná barva a neprůhlednost, *color* je aktuální barva a neprůhlednost vzorku, které ovlivňujeme na řádcích 2 a 3. Zde je *gradientMag* velikost gradientu, *sp* je výsledek stínování pro příslušný vzorek, *distToCamera* je vzdálenost od kamery. Parametry *kappa_t* a *kappa_s* ovlivňují vzdálenost pomyslné ořezové roviny a ostrost okrajů řezu.

3.3.5 Dvojúrovňové vykreslování

U dvojúrovňového vykreslování dat nejprve specifikujeme segmentaci dat. K tomu slouží posuvník *Object layers*, který můžeme vidět na Obrázku 3.5. Ten rozdělí objekt na x vrstev dle hodnoty skalárních dat. Každá vrstva nese své ID, díky kterému přiřazujeme různé metody vykreslení vrstvám. K určení metody uživatelem slouží výběrové pole níže. Poslední oddíl tvoří specifikace transfer funkce, kterou lze měnit pro každou vrstvu zvlášť. Záložky nalevo vybírají příslušnou vrstvu, která bude modifikována. Při změně metody je tato skutečnost zaznamenána do textury `methodIdTexture` na grafické kartě. Tuto texturu si předáme obdobně jako texturu objemovou či texturu transfer funkce do shaderu a můžeme za běhu upravovat její hodnoty. Tyto hodnoty určují následné skládání lokální úrovně. Pro složení vzorků na globální úrovni je třeba indikovat přechod mezi segmenty. Toho je dosaženo proměnnými *method* a *previousMethod*, které nesou informace o aktuální použité metodě vykreslování a o předchozí metodě. Při detekci přechodu dvou segmentů skládáme kompozičním



Obrázek 3.5: Uživatelské rozhraní pro dvojúrovňové vykreslování.

schématem akumulované hodnoty barvy a neprůhlednosti a počítáme s nimi jako s barvou a neprůhledností vzorku. Následně akumulované hodnoty vynulujeme a pokračujeme v lokální kompozici dalšího vzorku. Ten bude v budoucnu opět složen globálním kompozičním schématem. Výstupem fragment shaderu `dicom_shader_data_data.frag` je barva a neprůhlednost komponovaná globálně.

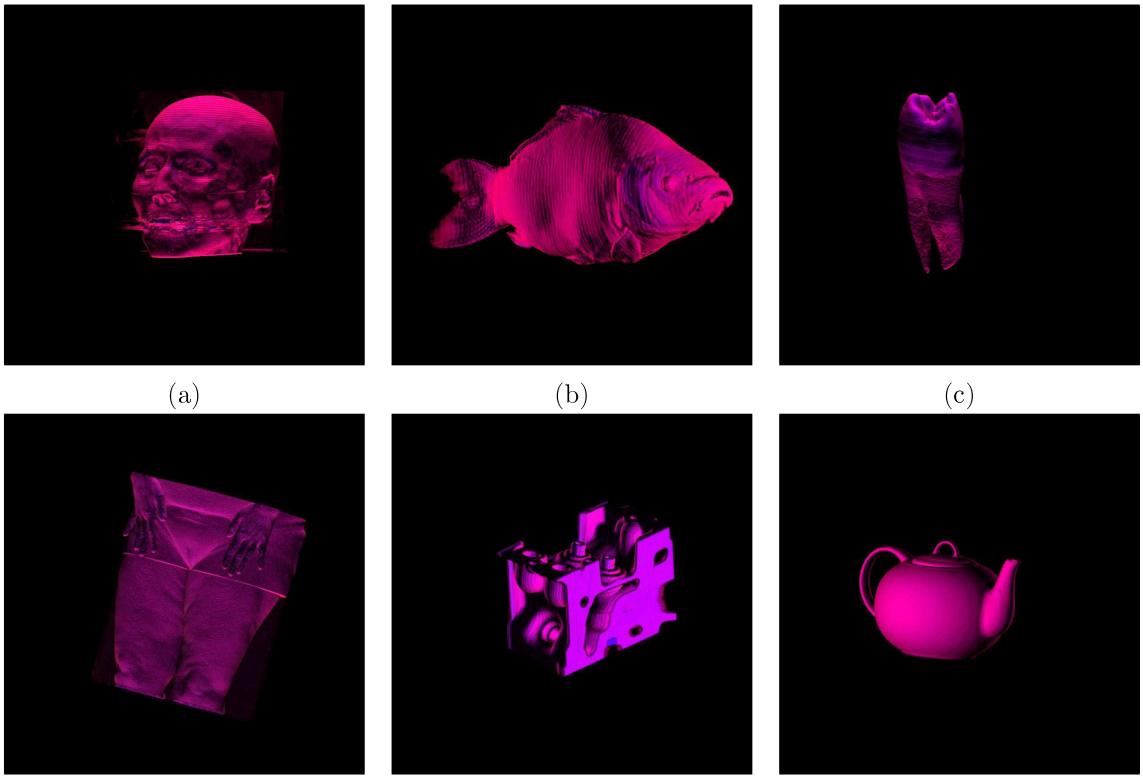
Kapitola 4

Testování

V této kapitole jsou představeny výsledky a měření implementovaných algoritmů pro vykreslování volumetrických dat společně s jejich porovnáním. Všechna měření jsou provedena na operačním systému Windows 10 s následující konfigurací systému:

- **Procesor:** Intel Core^{TM} i7-4510U CPU @ 2.00GHz, 2001 Mhz, jádra: 2
- **Paměť:** 2 x 4GiB DDR3 Hyundai Electronics 800 MHz
- **GPU:** NVIDIA GeForce GT 750M, 2048 MB GDDR5 @ 64.2 Gbps
- **Operační systém:** Windows 10 Home 64-bit
- **Java:** JDK 1.7
- **OpenGL:** OpenGL verze 3.0
- **GLSL:** verze 3.30 kompatibilní

Celkem bylo použito šest volumetrických modelů pro měření vlastností vykreslovacích metod. Tyto vlastnosti jsou jak výkonnostní, ale převážně vizuální, a budou tedy i vizuálně porovnávány. Všechna měření byla provedena na obrázcích s rozměry 800 $\times$ 800 pixelů (není-li uvedeno jinak). Přehled testovaných modelů můžeme vidět na Obrázku 4.1.



(a) (b) (c)

(d) (e) (f)

Obrázek 4.1: Přehled testovaných volumetrických modelů. (a) Model hlavy v rozměrech 256×256×113 pixelů získaný snímáním hlavy pomocí výpočetní tomografie, (b) Model ryby v rozměrech 256×256×512 pixelů, (c) Model zubu v rozměrech 256×256×161 pixelů, (d) Model pánevní oblasti a stehen ve formátu Dicom s rozměry 512×512×973 pixelů, (e) Model motoru v rozměrech 256×256×256 pixelů, (f) Model Bostonské čajové konvice skrývající humra s rozlišením 128×128×128 pixelů.

4.1 Výkonnostní srovnání

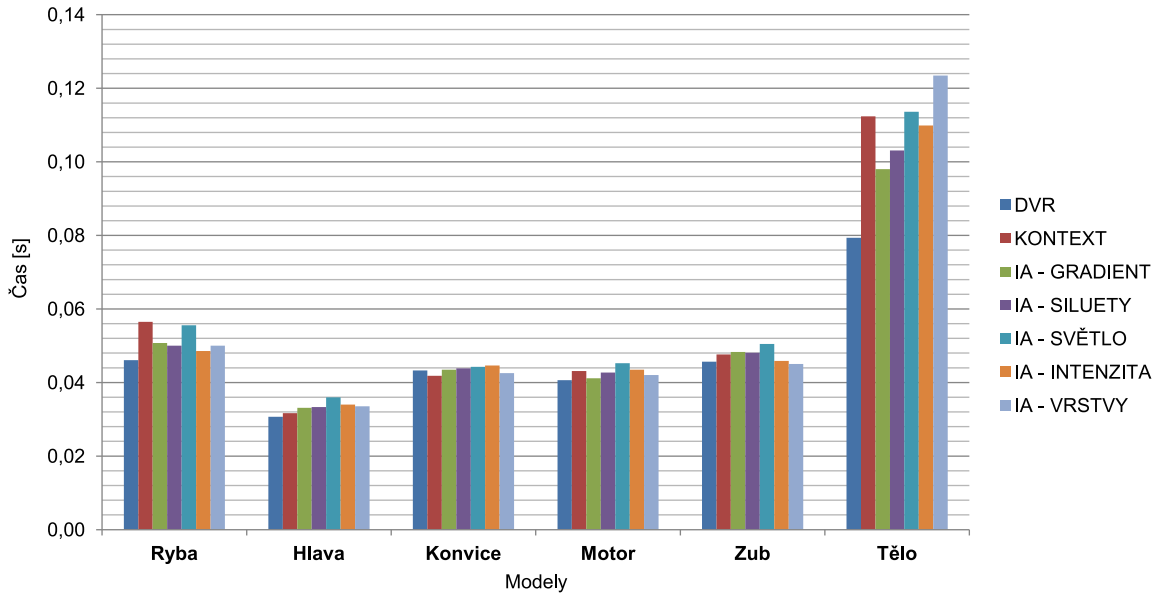
V Tabulce 4.1 jsou uvedeny naměřené hodnoty zobrazení snímků za sekundu (FPS) a doby vykreslení snímku jednotlivých modelů pro implementované algoritmy. K měření jsem využil vestavěné funkce knihovny *Tiger* pro určení FPS. Jednotlivá měření byla uskutečněna ze tří pohledů a zprůměrována z deseti měření pro každý pohled. Snímková frekvence je zaokrouhlena na jedno desetinné místo, čas vykreslení snímku je zaokrouhlen na tři desetinná místa. Z hodnot vyplývá, že nejméně snímků za sekundu dosahuje model Těla, což je způsobeno vyššími rozměry mřížky 512x512x973 pixelů. Z algoritmů jsou nejnáročnější na výkon metoda vykreslování zachovávající kontext, v tabulce označená jako **Kontext** a dále důležitostí řízené vykreslování, které důležitost vypočítává z osvětlovacího modelu, v tabulce označené jako **IA - Světlo**.

Graf 4.2 ukazuje srovnání rychlostí vykreslení jednotlivých metod pro testované modely. Očekávané rychlosti přímé metody (z předchozích měření v bakalářské práci) jsou v mnohých případech až překvapivě vyrovnány implementovanými algoritmy. Vyšších hodnot času

	DVR		Kontext		IA - Gradient		IA - Siluety		IA - Světlo		IA - Intenzita		IA - Vrstvy	
Model	**FPS**	**Čas [s]**	**FPS**	**Čas [s]**	**FPS**	**Čas [s]**	**FPS**	**Čas [s]**	**FPS**	**Čas [s]**	**FPS**	**Čas [s]**	**FPS**	**Čas [s]**
Ryba	21,7	0,046	17,7	0,056	19,7	0,051	20	0,050	18	0,056	20,6	0,049	20	0,050
Hlava	32,6	0,031	31,6	0,032	30,2	0,033	30	0,033	27,8	0,036	29,4	0,034	29,8	0,034
Konvice	23,1	0,043	23,9	0,042	23	0,043	22,8	0,044	22,6	0,044	22,4	0,045	23,5	0,043
Motor	24,6	0,041	23,2	0,043	24,3	0,041	23,4	0,043	22,1	0,045	23	0,043	23,8	0,042
Zub	21,9	0,046	21	0,048	20,7	0,048	20,8	0,048	19,8	0,051	21,8	0,046	22,2	0,045
Tělo	12,6	0,079	8,9	0,112	10,2	0,098	9,7	0,103	8,8	0,114	9,1	0,110	8,1	0,123

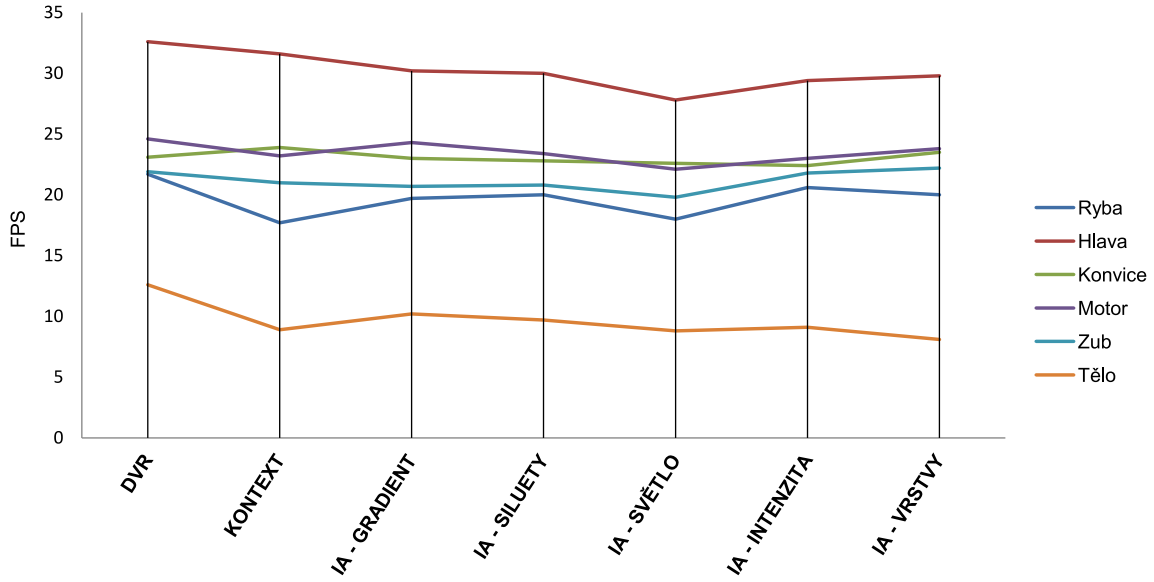
Tabulka 4.1: Tabulka měřených hodnot pro vybrané modely.

vykreslení snímku dosahují zpravidla již zmíněně metody vykreslení zachovávající kontext a důležitostí řízené vykreslování získávající důležitost z osvětlovacího modelu. Zajímavého trendu dosahují výsledky měření metody řízené důležitostí, která tento parametr přiřazuje pevnému počtu vrstev objektu. Pro modely s nižší datovou náročností je rozdíl přímé a této ilustrativní metody, ve srovnání s ostatními metodami, viditelně nižší, než-li například u modelu Těla.



Obrázek 4.2: Srovnání rychlostí vykreslení jednotlivých metod pro testované modely.

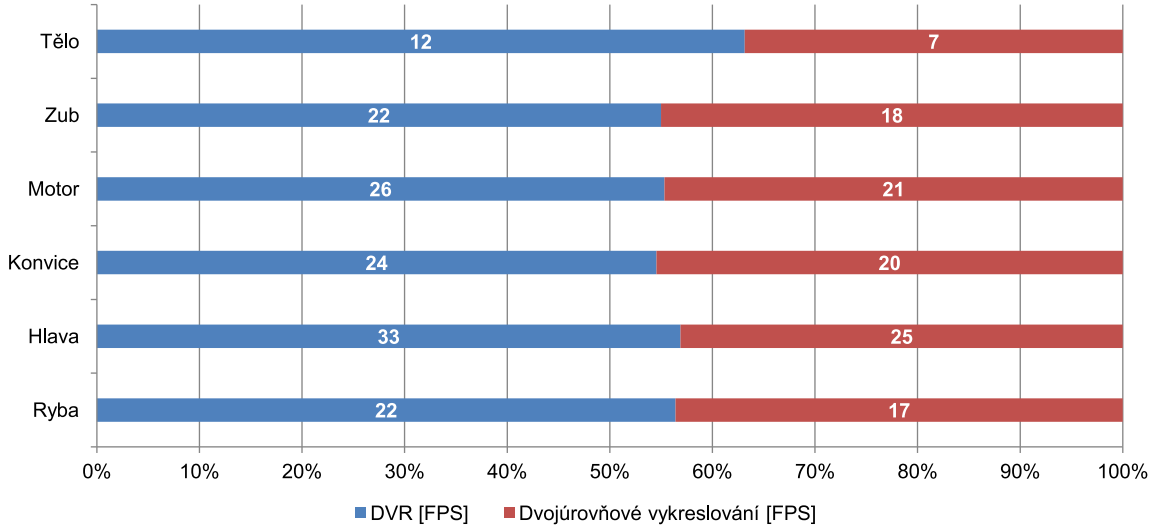
Graf 4.3 srovnává snímkovou frekvenci algoritmů v rovnoběžných souřadnicích. Z grafu je patrné, že zatížení metod na snímkovou frekvenci, a tedy i na výkon, není nijak zásadní. Nejlepší frekvence dosahuje přímá metoda. Výkon jednotlivých modelů odpovídá velikosti dat.



Obrázek 4.3: Srovnání FPS pro testované modely s různými metodami vykreslení.

Dvojúrovňové vykreslování bylo otestováno pro deset segmentovaných vrstev. Na každou z vrstev bylo aplikováno přímé vykreslování modelu. Měření bylo provedeno pro jeden pohled a bylo zprůměrováno z deseti hodnot. Graf 4.4 ukazuje dopad na snímkovou frekvenci při použití dvojúrovňového vykreslování. Frekvence opět závisí na velikosti mřížky skalárních dat, ovšem nemá tak vysoký vliv jako v předchozích případech. Značně důležitější je u této statistiky rozsah a rozložení skalárních hodnot. Modely, jejichž rozsah hodnot se často mění po celé své délce, jsou ovlivněny více, než-li modely s malým rozsahem hodnot. Příkladem je model Hlavy, který nese informace o vzduchu obklopujícím hlavu, kůži, svalstvu a vnitřní struktuře lebky - mozek a cévy. Tím pádem každá z vrstev přispívá do vykreslování a dopad na výkon je větší. Na rozdíl model Konvice, který je rozměry menším modelem, než-li model Hlavy, obsahuje takové signifikantní změny pouze dvě. Jedná se o vrstvu konvice a dále to je humr ukrytý uvnitř této konvice. Do vykreslování přispívá méně vrstev a proto je dopad na změnu výkonu menší.

Obecně lze konstatovat, že dvojúrovňové vykreslování má malý vliv na výkon, jak můžeme pozorovat v Grafu 4.4 a tento vliv je ovlivněn velikostí modelu a rozložením dat v jeho mřížce.



Obrázek 4.4: Podílové porovnání FPS pro dané modely s dvěma metodami vykreslení. Modře je znázorněna přímá metoda vykreslení. Červeně je znázorněna metoda dvojúrovňového vykreslení s deseti vrstvami. Na tyto vrstvy byla aplikována přímá metoda.

4.2 Vizuální srovnání

Vizuální srovnání je hlavní metrika pro výstup vizualizačních algoritmů. V této sekci jsou prezentovány a srovnány některé z výstupů v podobě obrázků. Dále pro připomenutí uvádím seznam parametrů použitých v této diplomové práci k lepšímu pochopení následujících obrázků:

- κ_t odpovídá hloubce umístění ořezové roviny u metody zachovávající kontext
- κ_s umožňuje kontrolu nad ostrostí řezu u stejnojmenné metody
- v_x reprezentuje hodnotu důležitosti přiřazené vrstvě x
- p_{l_1} ovlivňuje spekulární odraz u vykreslování s důležitostí získanou z osvětlení
- p_{l_2} ovlivňuje difúzní odraz u vykreslování s důležitostí získanou z osvětlení
- p_{l_3} ovlivňuje velikost gradientu u vykreslování s důležitostí získanou z osvětlení
- p_{s_1} parametr ovlivňující velikost gradientu u vykreslování s důležitostí siluet
- p_{s_2} parametr definující počátek sklonu funkce *smoothstep* u vykreslování siluet
- p_{s_3} parametr definující konec sklonu funkce *smoothstep* u vykreslování siluet
- *IA* je označení metody založené na vykreslování ovlivněném důležitostí (z angl. Importance-aware)

4.2.1 Vykreslování dat zachovávající kontext

Na Obrázku 4.5 vidíme čtveřici snímků, kde jsou první tři modely vykresleny pomocí ilustrativní metody vykreslování zachovávající kontext. Snímek 4.5a obsahuje konfiguraci parametrů $\kappa_s = 0.1$, $\kappa_t = 10$, kde první z parametrů ovlivňuje přechod okrajů ořezové roviny. Na tomto snímku vidíme více z vnitřku modelu, než-li na snímku následujícím 4.5b. Snímek 4.5c byl vykreslen s rozdílnou transfer funkcí, kde díky tomuto mapování maximalizujeme dopad metody zachování kontextu. Těleso zobrazuje velké množství informace z vnitřku objemu a přitom neztrácíme kontext. Poslední snímek 4.5d je pro srovnání vykreslen přímou metodou využívající stejné mapování 2D transfer funkcí jako u prvních dvou snímků, informace o vnitřku tělesa je ovšem v tomto případě skrytá.

(a) $\kappa_s = 0.1$, $\kappa_t = 10$

(b) $\kappa_s = 1$, $\kappa_t = 10$

(c) $\kappa_s = 0.1$, $\kappa_t = 11.25$

(d) DVR

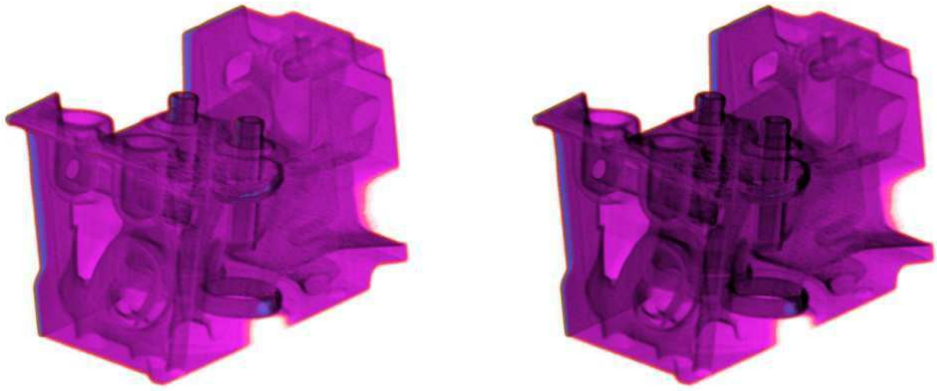
Obrázek 4.5: Vliv parametrů při vykreslování zachovávající kontext. Snímky mají invertované barvy.

Následující Obrázek 4.6 zobrazuje dva modely. Model konvice demonstruje efekt prvního parametru κ_s, kde okraje konvice u snímku 4.6b mají strmý přechod a jsou vykresleny méně,

oproti snímku 4.6c. Tyto konfigurace ovšem nejsou optimální, jelikož pouze snímek 4.6a nese informaci o orientaci konvice.

Model motoru na snímcích 4.6d a 4.6e potvrzuje správnost použití druhého parametru, který ovlivňuje hloubku (pomyslné) ořezové roviny tělesa. Při zachování přechodu okrajů je patrnější vnitřní struktura u druhého snímku.

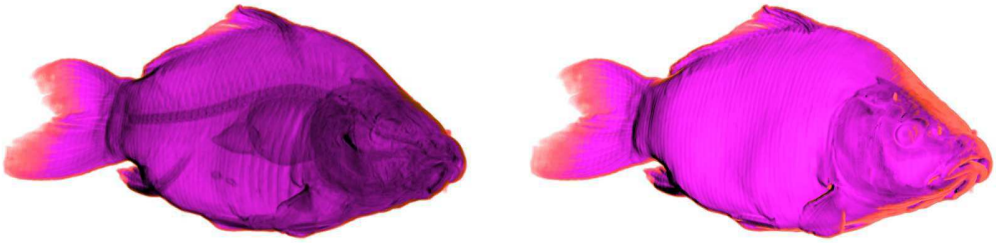
(a) $\kappa_s = 0.1$, $\kappa_t = 0.1$ (b) $\kappa_s = 0.1$, $\kappa_t = 10$ (c) $\kappa_s = 0.5$, $\kappa_t = 10$



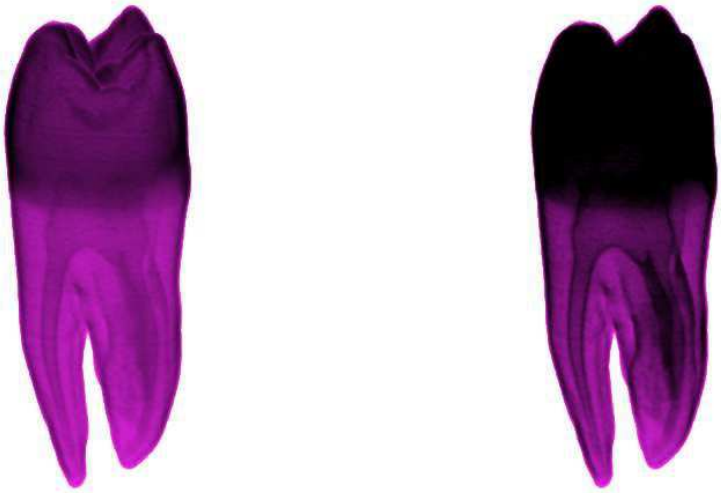
(d) $\kappa_s = 0.1$, $\kappa_t = 3.75$ (e) $\kappa_s = 0.1$, $\kappa_t = 15$

Obrázek 4.6: Vliv parametrů při vykreslování zachovávající kontext. Snímky mají invertované barvy.

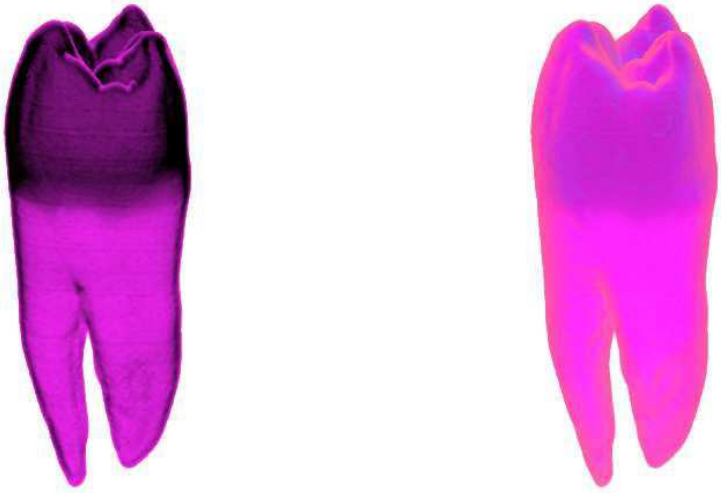
Další tři modely, využívající stejnou metodu, můžete pozorovat na Obrázcích 4.7 a 4.8. Druhý z Obrázků obsahuje snímky porovnatelné s výsledky autorů algoritmu (Obrázek 2.15), kde totožně vykreslujeme model hlavy. Vyšší hodnoty parametru κ_s vyústí v měkké okraje. Metoda zobrazí data umístěná dále od kamery s nárůstem parametru κ_t, velmi dobře viditelné na snímcích 4.7c, 4.7d. Pro každý model byla použitá stejná transfer funkce pro validní porovnání výsledků.



(a) $\kappa_s = 0.1$, $\kappa_t = 10$ (b) $\kappa_s = 1$, $\kappa_t = 10$

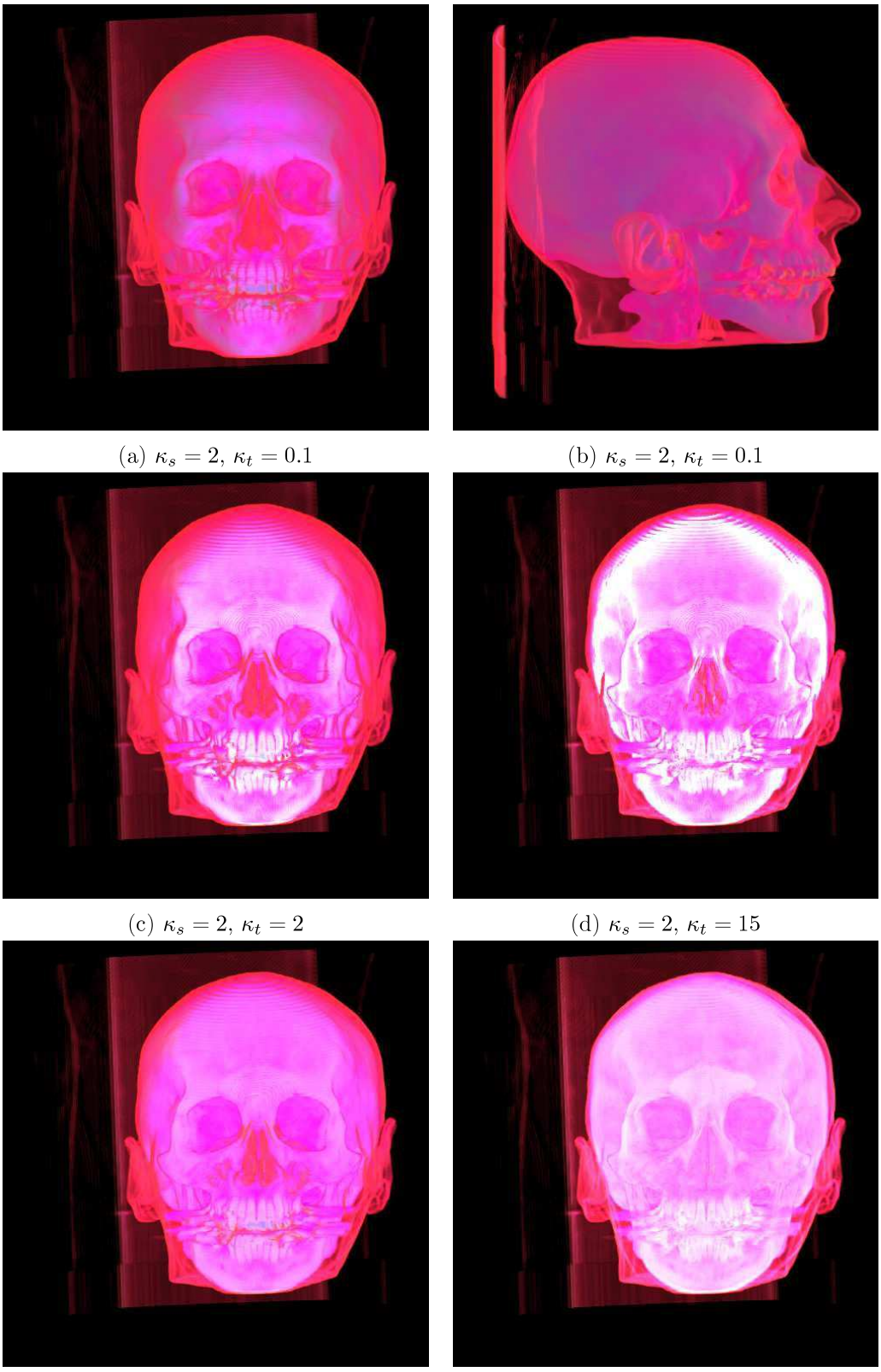


(c) $\kappa_s = 0.1$, $\kappa_t = 10$ (d) $\kappa_s = 1$, $\kappa_t = 10$



(e) $\kappa_s = 0.5$, $\kappa_t = 10$ (f) $\kappa_s = 2$, $\kappa_t = 0.1$

Obrázek 4.7: Vliv parametrů při vykreslování zachovávající kontext. Snímky mají invertované barvy.



(a) $\kappa_s = 2$, $\kappa_t = 0.1$

(b) $\kappa_s = 2$, $\kappa_t = 0.1$

(c) $\kappa_s = 2$, $\kappa_t = 2$

(d) $\kappa_s = 2$, $\kappa_t = 15$

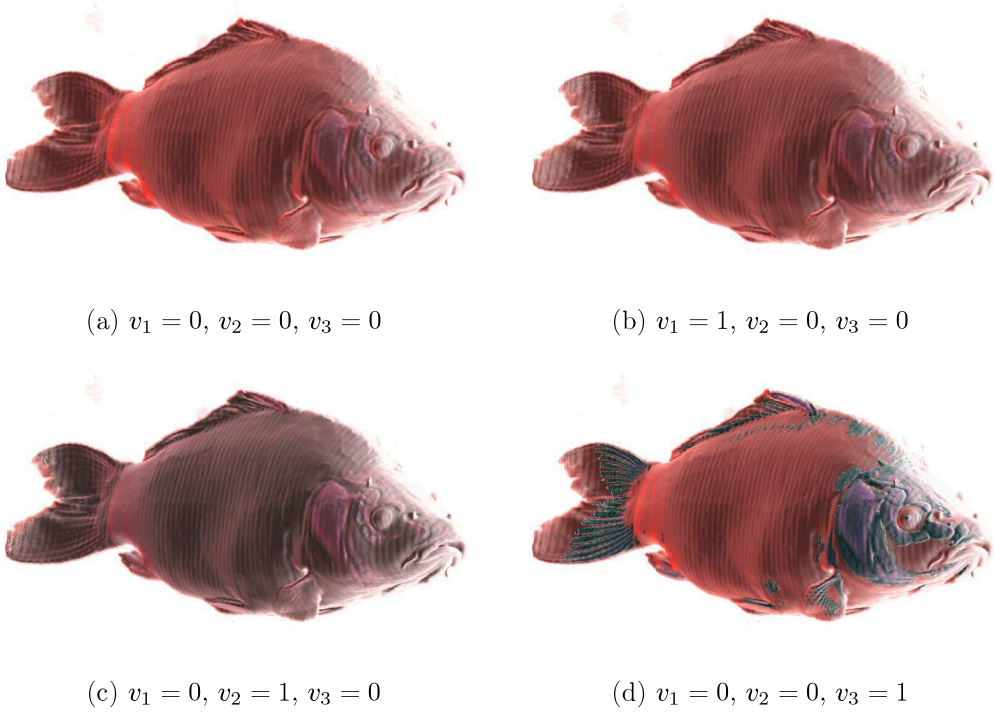
(e) $\kappa_s = 0.5$, $\kappa_t = 0.1$

(f) $\kappa_s = 0.25$, $\kappa_t = 0.1$

Obrázek 4.8: Vliv parametrů při vykreslování zachovávající kontext.

4.2.2 Vykreslování dat řízené důležitostí

V této sekci jsou prezentovány výsledky metody řízené důležitostí vzorků. Důležitost je získávána ze samotných dat více způsoby. První Obrázek 4.9 ukazuje čtyři snímky modelu ryby, kde byla důležitost přiřazena třem vrstvám. Tyto vrstvy byly vybrány na základě pevně určených intervalů hodnot skalárních dat. Vrstvy odpovídají měkkým částem ploutví, tělu a kostem. První snímek zobrazuje model s nastavením parametrů, kdy žádná vrstva nemá přiřazenou důležitost. Jedná se tedy o klasické vykreslení přímou metodou. Druhý snímek 4.9b vykreslí primárně měkké části ploutví, jejichž důležitost je rovna jedné. U druhého snímku vidíme tělo kapra s jiným odstínem barvy, jelikož je důležitost přiřazena tělu, ovšem vnější části odpovídající původnímu odstínu ve transfer funkci, jsou zanedbány. Nejzajímavějšího efektu je dosaženo ve čtvrtém snímku 4.9d. Kosti, odlišeny modře, jsou viditelné bez ořezové roviny či použití transfer funkce s nízkou hodnotou neprůhlednosti.



(a) $v_1 = 0, v_2 = 0, v_3 = 0$ (b) $v_1 = 1, v_2 = 0, v_3 = 0$

(c) $v_1 = 0, v_2 = 1, v_3 = 0$ (d) $v_1 = 0, v_2 = 0, v_3 = 1$

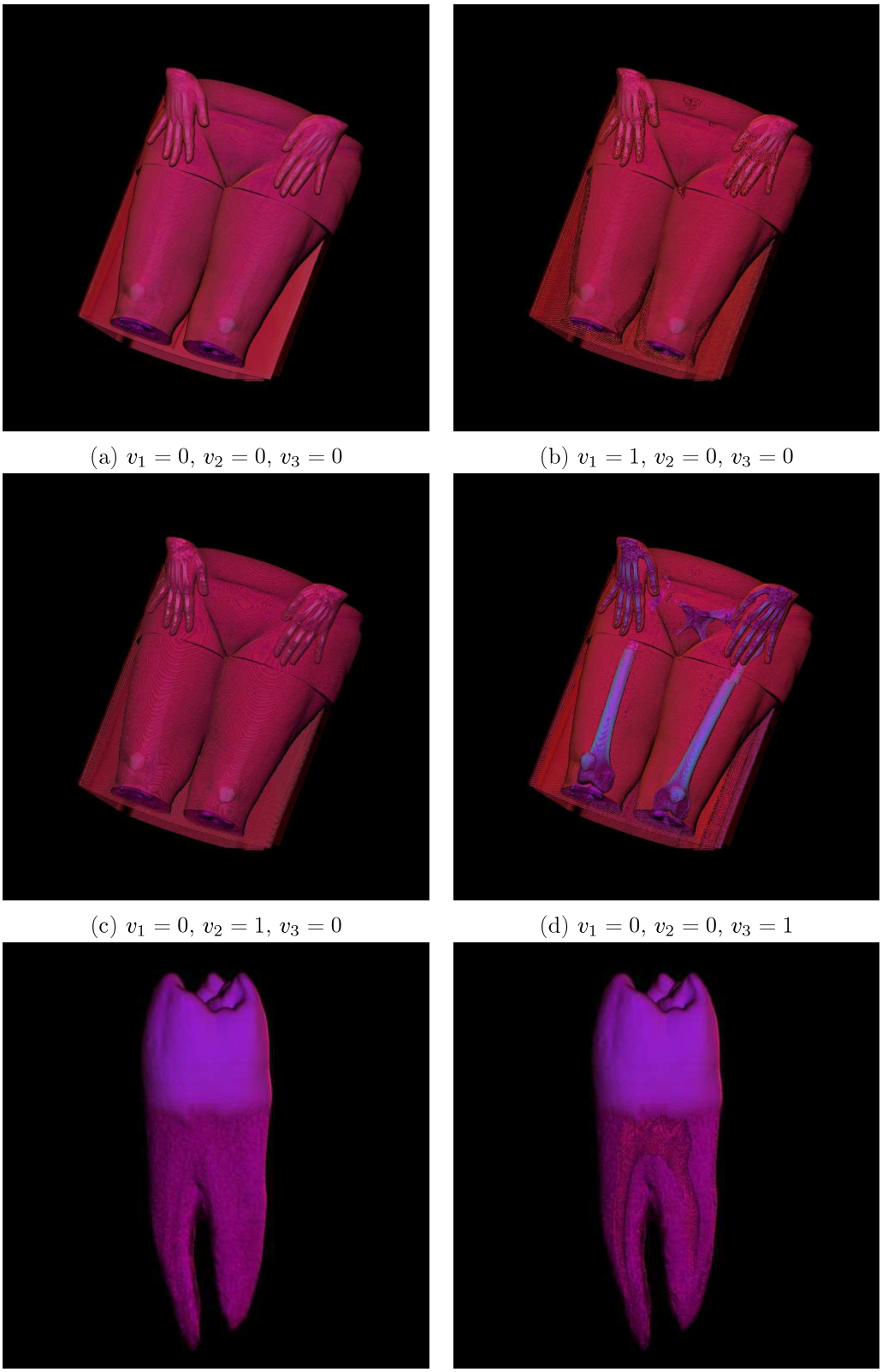
Obrázek 4.9: Vliv důležitosti vrstev u vykreslování řízeném důležitostí. Snímky mají invertované barvy.

Obrázky 4.10 a 4.11 ukazují stejné využití metody na třech dalších modelech. První ze série snímků modelu ukazuje metodu při vykreslení neovlivněném důležitostí. Následuje vrstva, kde přiřazujeme důležitost vzorkům s nízkými hodnotami dat. Zajímavá je na snímku 4.10f viditelná část zubu odpovídající kostní dřeni společně s nervy. Model lebky pak zobrazuje na snímku 4.11d lebeční dutiny viditelné na čele. Po přiřazení důležitosti prostřední vrstvě dosahujeme markantního výsledku u modelu hlavy na snímku 4.11e, kde je důležitost přiřazena kostem. Zuby, které mají vyšší hustotu, jsou pak vykresleny na následujícím snímku 4.11e společně s nadočnicovým obloukem a nejtvrdší částí dolní čelisti. Model zubu v posledním

snímku 4.11b zobrazuje nejtvrdší zubní vrstvu - sklovinu.

Obrázek 4.12 ukazuje model konvice, kde bylo využito osvětlení k získání důležitosti vzorků. Na druhém snímku 4.12b můžeme na rozdíl od snímku 4.12a pozorovat ukrytého humra, jehož zobrazení bylo docíleno modulováním parametru p_{l_3} ovlivňující velikost gradientu.

Poslední Obrázek 4.13 ukazuje účinek stejné metody, kde byla důležitost přiřazena na základě velikosti gradientu. Prvky motoru s vyššími hodnotami potlačí nedůležité části, což nám umožní vhled do motoru, aniž bychom měnili transfer funkci.



(a) $v_1 = 0$, $v_2 = 0$, $v_3 = 0$

(b) $v_1 = 1$, $v_2 = 0$, $v_3 = 0$

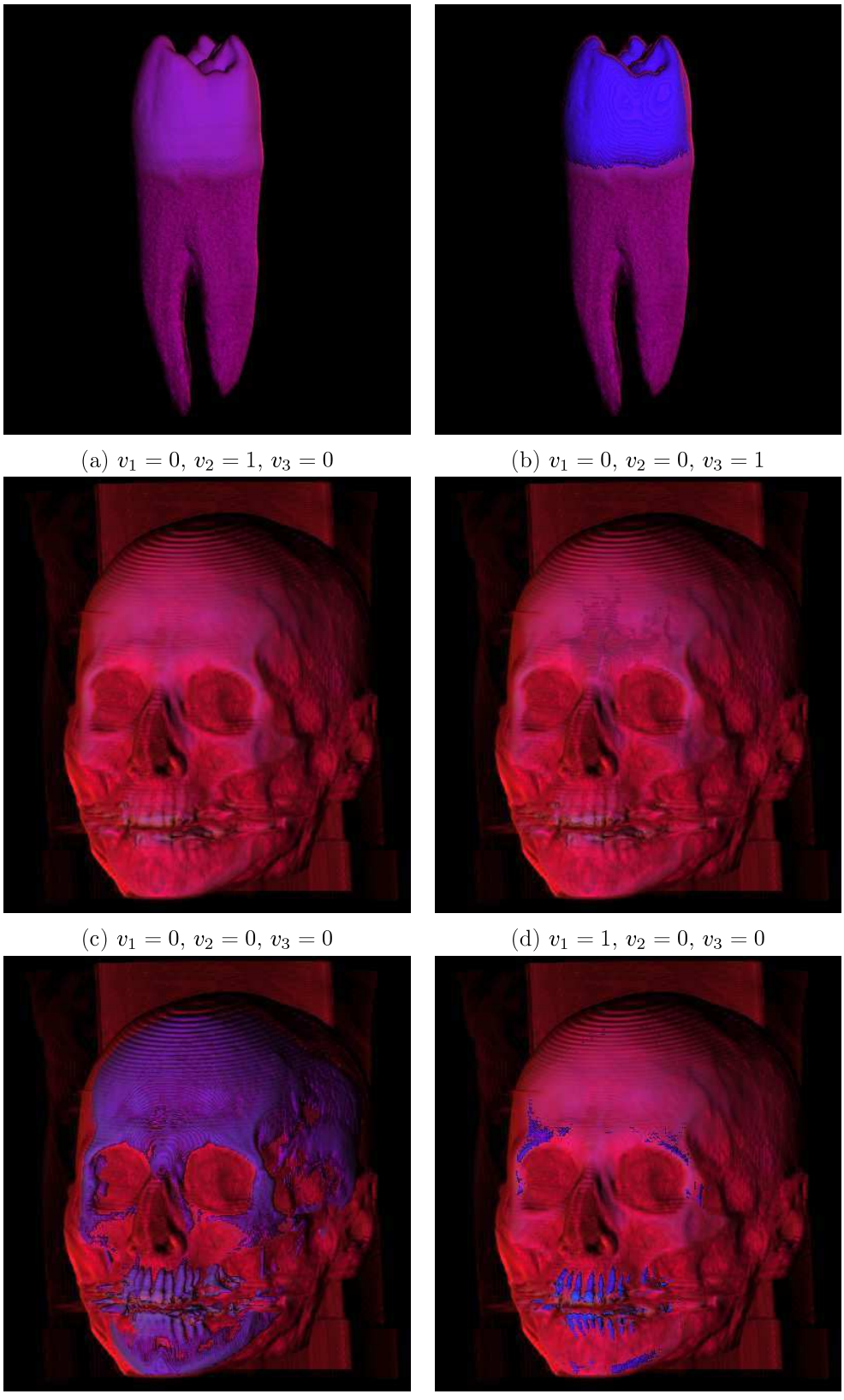
(c) $v_1 = 0$, $v_2 = 1$, $v_3 = 0$

(d) $v_1 = 0$, $v_2 = 0$, $v_3 = 1$

(e) $v_1 = 0$, $v_2 = 0$, $v_3 = 0$

(f) $v_1 = 1$, $v_2 = 0$, $v_3 = 0$

Obrázek 4.10: Vliv důležitosti vrstev u vykreslování řízeném důležitostí.



(a) $v_1 = 0$, $v_2 = 1$, $v_3 = 0$

(b) $v_1 = 0$, $v_2 = 0$, $v_3 = 1$

(c) $v_1 = 0$, $v_2 = 0$, $v_3 = 0$

(d) $v_1 = 1$, $v_2 = 0$, $v_3 = 0$

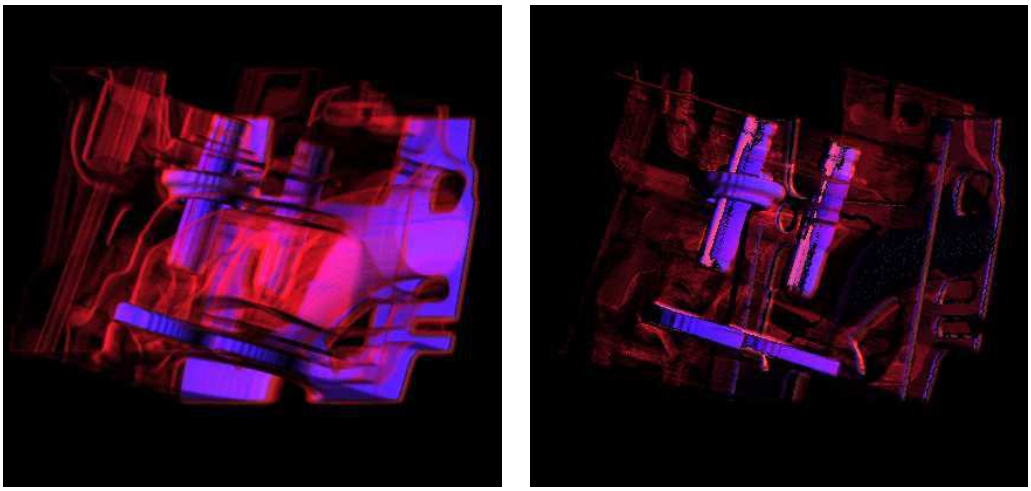
(e) $v_1 = 0$, $v_2 = 1$, $v_3 = 0$

(f) $v_1 = 0$, $v_2 = 0$, $v_3 = 1$

Obrázek 4.11: Vliv důležitosti vrstev u vykreslování řízeném důležitostí.

(a) $p_{l_1} = 75$, $p_{l_2} = 100$, $p_{l_3} = 100$ (b) $p_{l_1} = 75$, $p_{l_2} = 100$, $p_{l_3} = 0$

Obrázek 4.12: Vliv důležitosti u vykreslování řízeném důležitostí získané z osvětlení. Snímky mají invertované barvy.

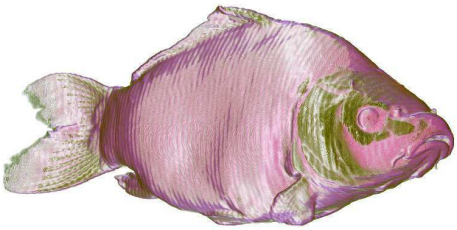


(a) 2D transfer funkce (b) IA - gradient

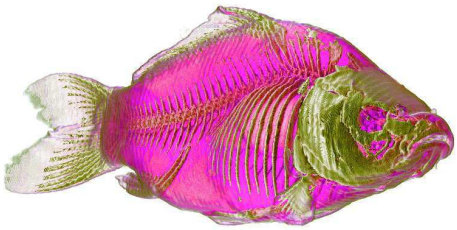
Obrázek 4.13: Vliv důležitosti u vykreslování řízeném důležitostí získané z velikosti gradientu.

4.2.3 Dvojúrovňové vykreslování segmentovaných dat

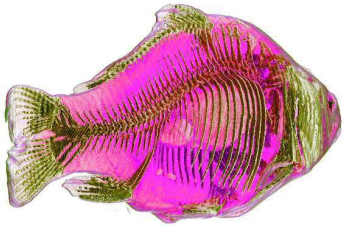
Kombinací implementovaných metod společně s přímým vykreslováním, metodou first hit a maximum intensity projection lze dosáhnout obrázků představených v této sekci. Připomínám, že pro každý model byla použita jedna transfer funkce, aby bylo možné přesně určit vizuální dopad jednotlivých metod. První dva Obrázky demonstrují kombinaci různých metod na modelu ryby. Velmi patrný je rozdíl snímků 4.14a, 4.14b. Na tělo ryby byla aplikována přímá metoda a metoda zachovávající kontext. Druhá z metod umožňuje při zachování kontextu nahlédnout dovnitř modelu a prozkoumat tak kosti a orgány ryby.



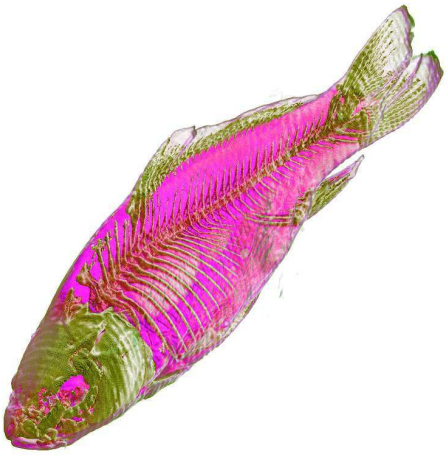
(a)



(b) $\kappa_s = 0.1$, $\kappa_t = 10$



(c) $\kappa_s = 0.1$, $\kappa_t = 10$

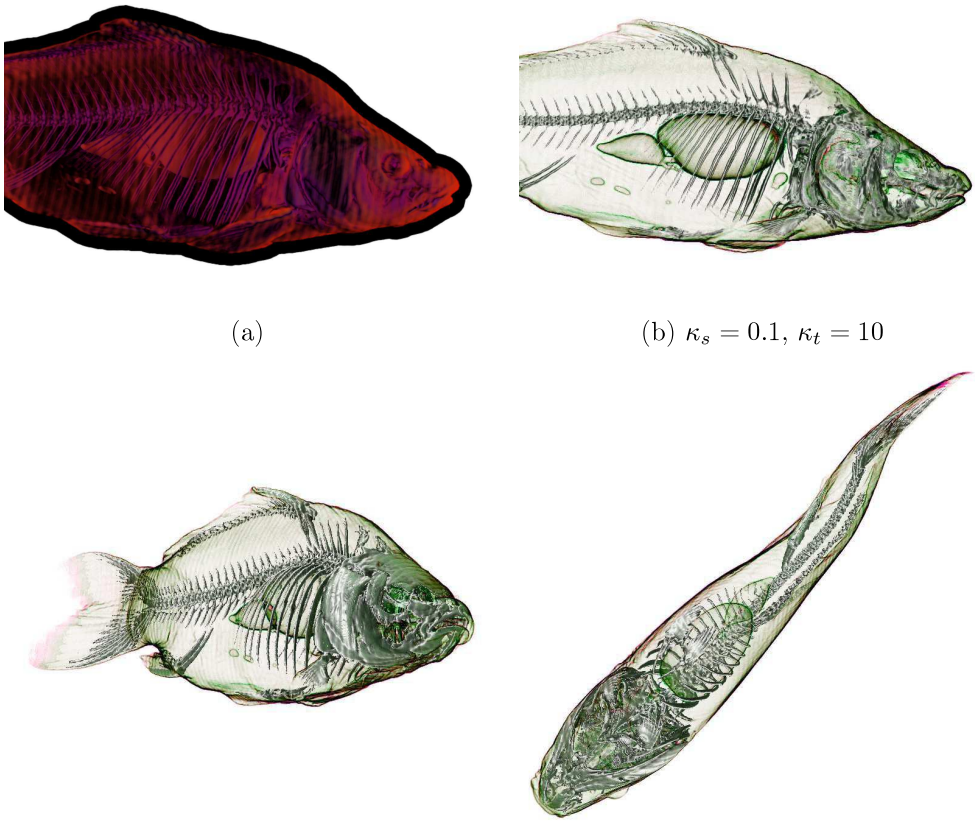


(d) $\kappa_s = 0.1$, $\kappa_t = 10$

Obrázek 4.14: Dvě kombinace metod vykreslení segmentovaných dat. (a) Tělo ryby je vykresleno pomocí přímého vykreslení dat, vrstva obsahující měkké kosti využívá důležitosti siluet, kosti s vyšší hustotou jsou vykresleny metodou first hit; (b), (c) a (d) Tělo ryby je vykresleno pomocí vykreslování zachovávající kontext, vrstva obsahující měkké kosti využívá důležitosti siluet, a kosti s vyšší hustotou jsou vykresleny stejně metodou first hit.

Podobného výsledku ovšem s jinou kombinací metod a jinou transfer funkcí bylo dosaženo na Obrázku 4.15. Vidíme, že dvojúrovňové vykreslování poskytuje více informace v přehlednější formě.

Obrázek 4.16 ukazuje 6 přístupů vykreslení dat. Snímek 4.16a vykresluje kůži metodou,

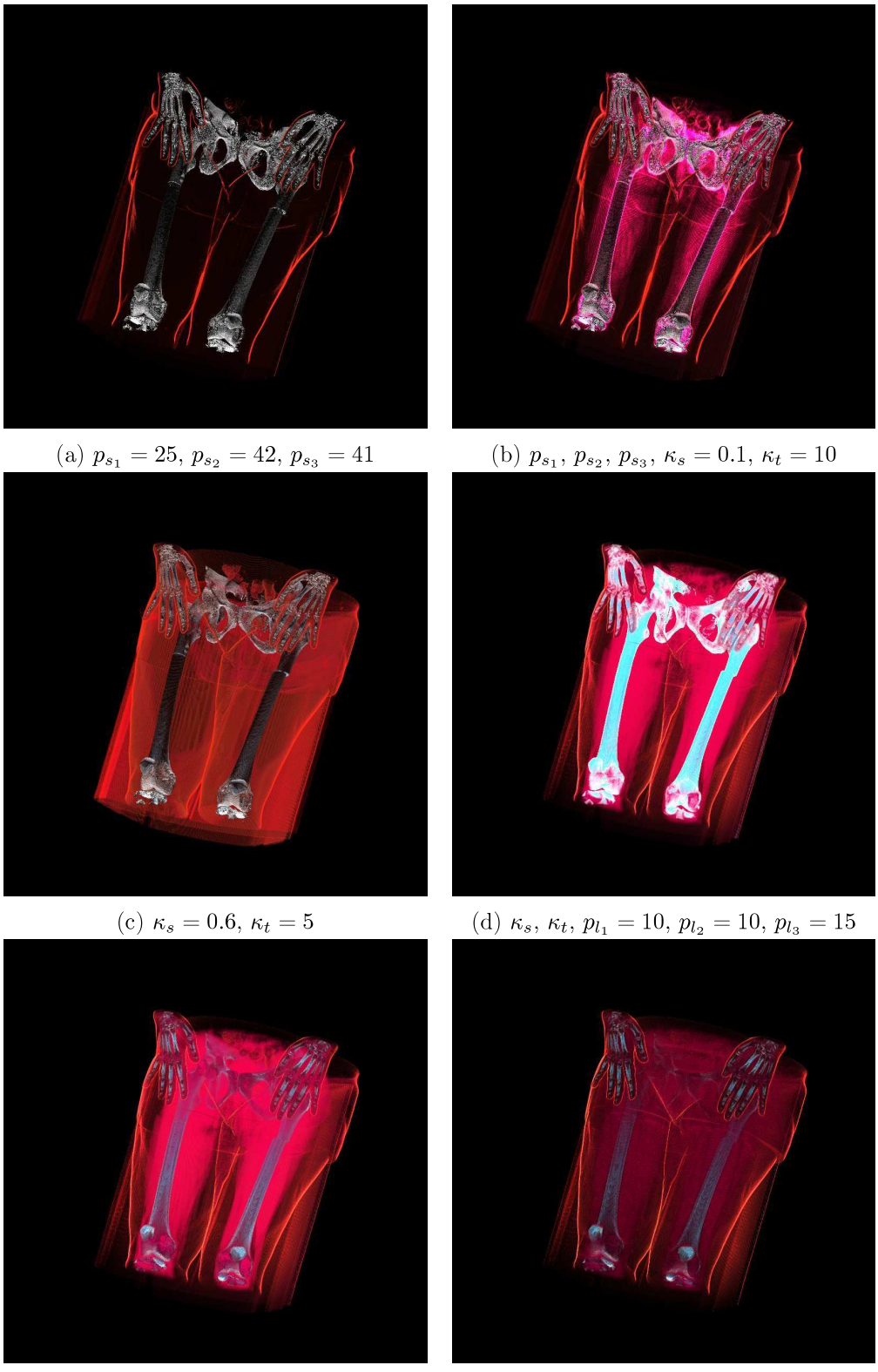


(a) (b) $\kappa_s = 0.1$, $\kappa_t = 10$

(c) $\kappa_s = 0.1$, $\kappa_t = 10$ (d) $\kappa_s = 0.1$, $\kappa_t = 10$

Obrázek 4.15: Porovnání přístupu vykreslení segmentovaných dat a přímého vykreslení. (a) Tělo ryby je vykresleno pomocí přímého vykreslení dat; (b), (c) a (d) Tělo ryby a vrstva obsahující měkké kosti jsou vykresleny pomocí vykreslování zachovávající kontext, kosti s vyšší hustotou jsou vykresleny metodou first hit. První snímek má oříznuté pozadí, poslední tři snímky mají invertované barvy.

kde je důležitost přiřazená siluetám objektu. Kosti jsou vykresleny metodou first hit. Snímek 4.16b pak přidává třetí vrstvu svalstva využívající metodu zachovávající kontext. Kosti jsou na druhém snímku lépe rozeznatelné. Snímky 4.16c - 4.16f se skládají ze tří vrstev. Vrstva kůže, vykreslená metodou zachovávající kontext a vrstva kostí, využívající důležitosti přiřazenou na základě intenzity dat, jsou pro všechny snímky stejné.



(a) $p_{s_1} = 25$, $p_{s_2} = 42$, $p_{s_3} = 41$

(b) p_{s_1}, p_{s_2}, p_{s_3}, $\kappa_s = 0.1$, $\kappa_t = 10$

(c) $\kappa_s = 0.6$, $\kappa_t = 5$

(d) κ_s, κ_t, $p_{l_1} = 10$, $p_{l_2} = 10$, $p_{l_3} = 15$

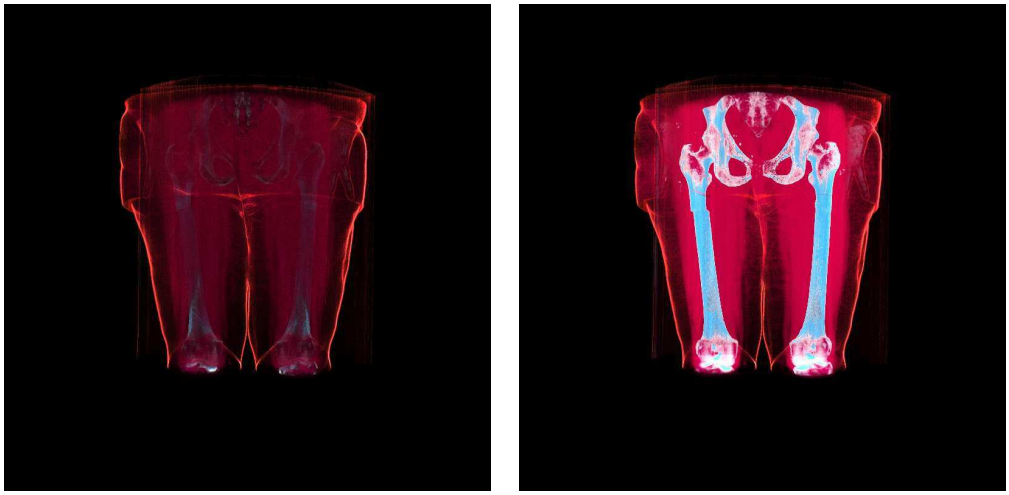
(e) κ_s, κ_t

(f) κ_s, κ_t

Obrázek 4.16: Šest kombinací metod vykreslení segmentovaných dat.

Prostřední vrstva svalstva je pro snímky určena následovně: 4.16c - maximum intensity projection, 4.16d - IA - Světlo, 4.16e - IA - gradient a transfer funkce pro poslední snímek 4.16f.

Porovnání přímé metody a metody s důležitostí na základě osvětlení můžeme pozorovat na Obrázku 4.17. Jedná se o pohled na zadní část modelu. Pro přímé vykreslení nejsou kosti prakticky vůbec viditelné. Druhý přístup dává vynikající výsledky i přes fakt, že mezi kostmi a pozorovatelem se nachází silná vrstva podložky, ne které žena ležela.



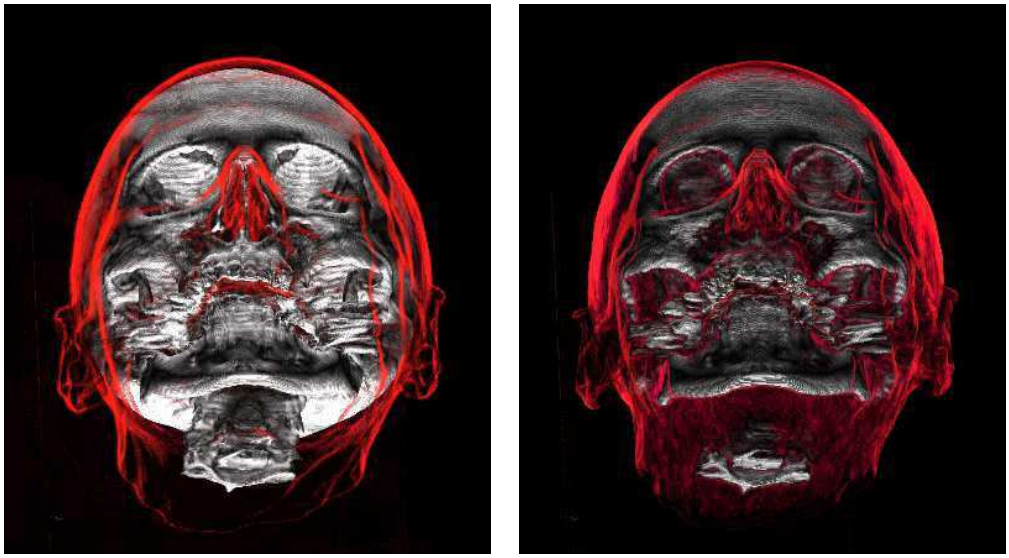
(a) $\kappa_s = 0.6$, $\kappa_t = 5$ (b) κ_s, κ_t, $p_{l_1} = 10$, $p_{l_2} = 10$, $p_{l_3} = 15$

Obrázek 4.17: Pohled zezadu na model, který byl vykreslen pomocí dvou různých kombinací metod vykreslení segmentovaných dat.

Obrázek 4.18 porovnává siluety kůže 4.18a a svalstva 4.18b. Zde dosahujeme velmi podobných výsledků.

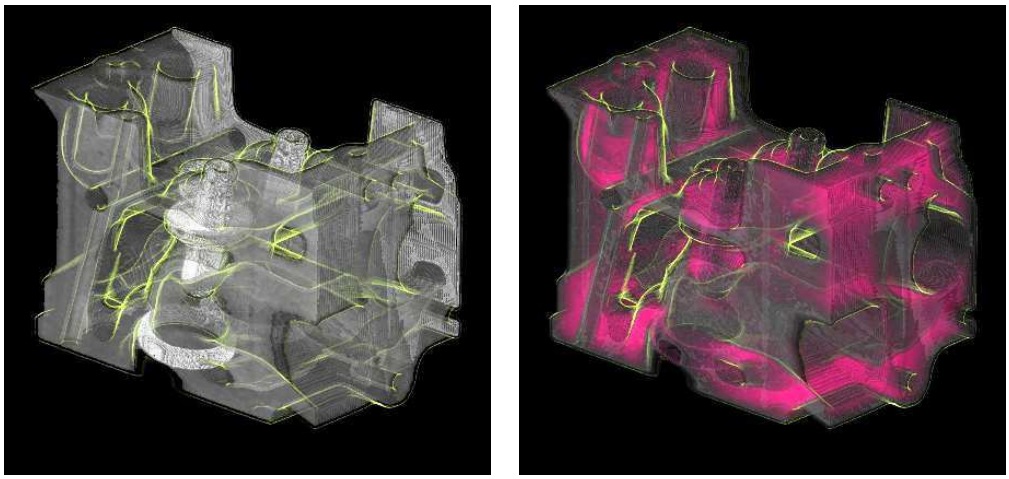
Poslední model zobrazuje motor s kombinací metod siluet a maximum intensity projection (snímek 4.19a) a dále model obohacený navíc o vrstvu vykreslenou díky důležitosti získanou z intenzity dat (snímek 4.19a). Vnímání kontextu plných částí motoru je díky této vrstvě lepší na druhém snímku.

Obrázek 4.20 prezentuje 3 metody vykreslení motoru ovlivněného transfer funkcí s pozvolným přechodem. Boční pohled v prvním snímku zobrazuje siluety, a zároveň vnitřní části, díky vykreslení zachovávající kontext. V druhém případě se jedná o přímé vykreslení, kde je rozdíl metod zjevný. V třetím případě byly vykresleny pouze siluety motoru. Zde nepatrně ztrácíme data ze zadní části, což má ovšem za následek lepší přehlednost.



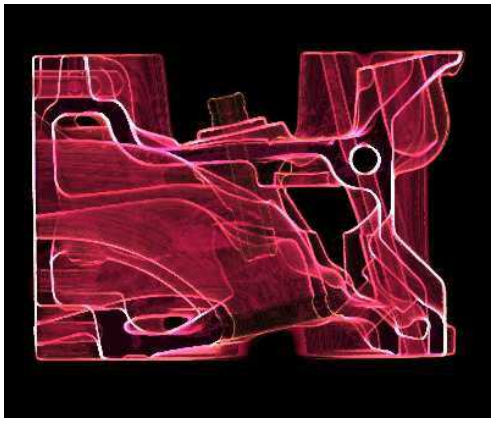
(a) (b)

Obrázek 4.18: (a) Vykreslení siluet kůže; (b) vykreslení siluet vrstvy obsahující svalstvo.

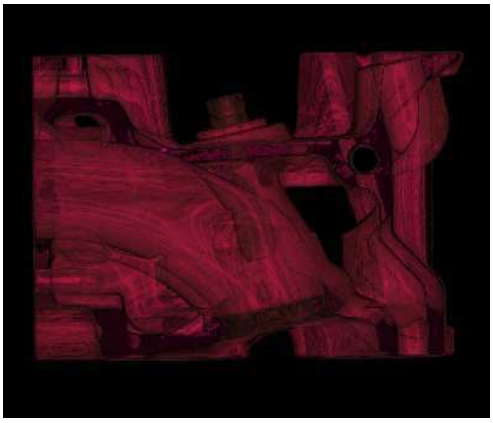


(a) $p_{s_1} = 20$, $p_{s_2} = 30$, $p_{s_3} = 28$ (b) $p_{s_1} = 20$, $p_{s_2} = 30$, $p_{s_3} = 28$

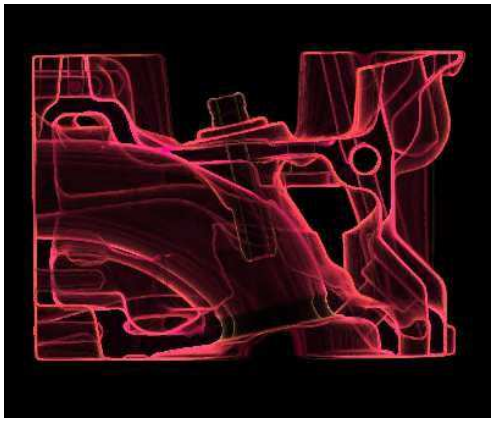
Obrázek 4.19: Ovlivnění vnímání kontextu plných částí motoru.



(a) $\kappa_s = 0.3$, $\kappa_t = 5$



(b) DVR



(c) $p_{s_1} = 35$, $p_{s_2} = 35$, $p_{s_3} = 1$

Obrázek 4.20: Tři kombinace metod vykreslení segmentovaných dat.

Kapitola 5

Závěr

V této práci jsem se zabýval problematikou vykreslování volumetrických dat, konkrétně metodou, která umožňuje kombinovat různé vizualizační techniky při akumulaci barvy a průhlednosti podél paprsku. Prezentoval jsem teoretický základ vykreslování volumetrických dat, vycházející z mé bakalářské práce. Ten jsem rozšířil o analýzu vykreslování segmentovaných dat a dvou nefotorealistických technik, tedy vykreslování volumetrických dat řízené důležitostí a ilustrativní vykreslování dat zachovávající kontext. Analyzoval jsem způsob dvojúrovňového vykreslování, pomocí kterého můžeme komponovat různé techniky a jejich různé kombinace.

Realizoval jsem segmentaci dat na základě skalárních hodnot mřížky. Dvě zmíněné nefotorealistické metody jsem, dle analýzy a doporučené literatury, implementoval a u druhé metody jsem umožnil různé způsoby měření důležitosti ze skalárních dat. Následně jsem tyto metody integroval, spolu se stávajícími metodami přímého vykreslení, MIP a First hit, do dvojúrovňového vykreslování, kde lze tyto metody přiřazovat jednotlivým úsekům volumetrického modelu a vzájemně je kombinovat.

Implementaci jsem otestoval na více než šedesáti scénách, patřících šesti modelům s mřížkami různého rozlišení. Výsledky jsem otestoval jak vizuálním srovnáním, tak výkonnostním měřením při rozlišení obrázků 800×800 pixelů.

Z testování vyplývá, že dvojúrovňové vykreslování má negativní vliv na výkon, který ovšem není markantní. Z implementovaných metod jsou obecně nejnáročnější na výkon metoda vykreslování zachovávající kontext a dále důležitostí řízené vykreslování, které důležitost vypočítává z osvětlovacího modelu. Očekávané rychlosti přímé metody (z předchozích měření v bakalářské práci) jsou v mnohých případech překvapivě vyrovnány (či pokořeny) implementovanými metodami. U vizuálního srovnání jsem demonstroval, jakých efektů lze dosáhnout pomocí ilustrativních technik, kde jsem srovnal jednotlivé přístupy a dále jsem prezentoval jejich různé kombinace pomocí dvojúrovňového vykreslování. Nefotorealistické metody a jejich kombinace poskytují možnosti vyobrazení vnitřních struktur a objektů zájmu bez ztráty kontextu.

5.1 Možné pokračování práce

Dvojúrovňové vykreslování je snadno rozšiřitelné o nové metody, které je pak možné kombinovat při akumulaci podél paprsku. Umožnění dalších kombinací společně s realizací nových přístupů pro globální kompozici se jeví jako vhodný kandidát pro budoucí práci. Tyto kombinace mohou vylepšit dosavadní vizualizace, či dát vzniku novým konceptům, které dosud nebyly objeveny.

Literatura

[1] Prezentace - Vizualizace, Direct Volume Rendering. <http://leyfi.felk.cvut.cz/courses/viz/lecture05/VIS-Modules-06-Direct_Volume_Rendering.pdf>. Datum: 2017-05-19.

[2] Prezentace - APG, Světlo. <https://cent.felk.cvut.cz/courses/APG/PDF/07_Svetlo.pdf>. Datum: 2017-01-17.

[3] Definice formátu DICOM. <https://cs.wikipedia.org/wiki/DICOM>. Datum: 2017-01-15.

[4] MRI - Magnetická rezonance. <https://en.wikipedia.org/wiki/Magnetic_resonance_imaging>. Datum: 2017-05-18.

[5] Ray Casting. <http://commons.wikimedia.org/wiki/File:Volume_ray_casting.png>. Datum: 2017-05-19.

[6] Reprezentace 3D těles. <https://leyfi.felk.cvut.cz/courses/mga/lectures/05_reprezentace_3d_teles.pdf>. Datum: 2017-05-19.

[7] Definition of the Definite Integral. <http://commons.wikimedia.org/wiki/File:Riemann-Sum-right-hand.png>. Datum: 2017-05-19.

[8] BRUCKNER, S. et al. Illustrative context-preserving volume rendering. In *EuroVis*, s. 69–76, 2005.

[9] CHEN, J. X. – WEGMAN, E. J. *Foundations of 3D Graphics Programming.* Springer, 2006.

[10] CSÉBFALVI, B. et al. Fast Visualization of Object Contours by Non-Photorealistic Volume Rendering. In *Computer Graphics Forum*, 20, s. 452–460. Wiley Online Library, 2001.

[11] DANSKIN, J. – HANRAHAN, P. Fast algorithms for volume ray tracing. In *Proceedings of the 1992 workshop on Volume visualization*, s. 91–98. ACM, 1992.

[12] MOURA PINTO, F. – FREITAS, C. M. Importance-aware composition for illustrative volume rendering. In *2010 23rd SIBGRAPI Conference on Graphics, Patterns and Images*, s. 134–141. IEEE, 2010.

[13] DREBIN, R. A. – CARPENTER, L. – HANRAHAN, P. Volume Rendering. *SIGGRAPH Comput. Graph.* June 1988, 22, 4, s. 65–74. ISSN 0097-8930. doi: 10.1145/378456.378484. Dostupné z: <http://doi.acm.org/10.1145/378456.378484>.

[14] DUDA, R. O. – HART, P. E. – STORK, D. G. *Pattern classification.* John Wiley & Sons, 2012.

[15] FERNANDO, R. – HAINES, E. – SWEENEY, T. GPU gems: programming techniques, tips, and tricks for real-time graphics. *Dimensions.* 2001, 7, 4, s. 816.

[16] Framebuffer. Framebuffer — Wikipedia, The Free Encyclopedia, 2015. Dostupné z: <http://en.wikipedia.org/wiki/Framebuffer>. [Online; Datum: 2017-05-19].

[17] GELDER, A. V. – KIM, K. Direct volume rendering with shading via three-dimensional textures. In *Proceedings of 1996 Symposium on Volume Visualization*, s. 23–30, 98, Oct 1996. doi: 10.1109/SVV.1996.558039.

[18] GLASSNER, A. S. *Principles of digital image synthesis.* Morgan Kaufmann, 2014.

[19] GROLL, J. Přímé vykreslování volumetrických dat na GPU. Bakalářská práce, České vysoké učení technické v Praze, 2015.

[20] HADWIGER, M. – BERGER, C. – HAUSER, H. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Visualization, 2003. VIS 2003. IEEE*, s. 301–308. IEEE, 2003.

[21] HADWIGER, M. et al. *Real-time Volume Graphics.* Natick, MA, USA : A. K. Peters, Ltd., 2006. ISBN 1568812663.

[22] HAUSER, H. et al. Two-level volume rendering. *IEEE transactions on visualization and computer graphics.* 2001, 7, 3, s. 242–252.

[23] HOWELL, J. R. – MENGUC, M. P. – SIEGEL, R. *Thermal radiation heat transfer.* : CRC press, 2010.

[24] Ing. Ladislav Čmolík, Ph.D. DCGI Ing. Ladislav Čmolík, Ph.D. Profile, 2015. Dostupné z: <http://dcgi.felk.cvut.cz/home/cmolikl/>. [Online; Datum: 2017-05-19].

[25] KAUFMAN, A. – MUELLER, K. Overview of volume rendering. *The visualization handbook.* 2005, 7, s. 127–174.

[26] KINDLMANN, G. Transfer functions in direct volume rendering: Design, interface, interaction. *Course notes of ACM SIGGRAPH.* 2002.

[27] KINDLMANN, G. – DURKIN, J. W. Semi-automatic generation of transfer functions for direct volume rendering. In *Proceedings of the 1998 IEEE symposium on Volume visualization*, s. 79–86. ACM, 1998.

[28] KINDLMANN, G. L. Semi-automatic generation of transfer functions for direct volume rendering. Diplomová práce, Citeseer, 1999.

[29] KNISS, J. – KINDLMANN, G. – HANSEN, C. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of the conference on Visualization'01*, s. 255–262. IEEE Computer Society, 2001.

[30] KRUEGER, W. Volume Rendering and Data Feature Enhancement. *SIGGRAPH Comput. Graph.* November 1990, 24, 5, s. 21–26. ISSN 0097-8930. doi: 10.1145/99308.99312. Dostupné z: <http://doi.acm.org/10.1145/99308.99312>.

[31] LACROUTE, P. – LEVOY, M. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, s. 451–458. ACM, 1994.

[32] LAUR, D. – HANRAHAN, P. Hierarchical splatting: A progressive refinement algorithm for volume rendering. In *ACM SIGGRAPH Computer Graphics*, 25, s. 285–288. ACM, 1991.

[33] LEVOY, M. Display of surfaces from volume data. *Computer Graphics and Applications, IEEE*. 1988, 8, 3, s. 29–37.

[34] LEVOY, M. A hybrid ray tracer for rendering polygon and volume data. *IEEE Computer graphics and Applications*. 1990, 10, 2, s. 33–40.

[35] LU, A. et al. Non-photorealistic volume rendering using stippling techniques. In *Visualization, 2002. VIS 2002. IEEE*, s. 211–218. IEEE, 2002.

[36] MAX, N. Optical models for direct volume rendering. *Visualization and Computer Graphics, IEEE Transactions on*. 1995, 1, 2.

[37] MAX, N. – HANRAHAN, P. – CRAWFIS, R. Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions. *SIGGRAPH Comput. Graph.* November 1990, 24, 5, s. 27–33. ISSN 0097-8930. doi: 10.1145/99308.99315. Dostupné z: <http://doi.acm.org/10.1145/99308.99315>.

[38] OpenGL. OpenGL — Wikipedia, The Free Encyclopedia, 2015. Dostupné z: <http://cs.wikipedia.org/wiki/OpenGL>. [Online; Datum: 2017-05-19].

[39] OpenGL Shading Language. OpenGL Shading Language Official Webpage, 2015. Dostupné z: <https://www.opengl.org/documentation/glsl/>. [Online; Datum: 2017-05-19].

[40] Petr Felkel. OpenGL jako automat - prezentace, 2017. Dostupné z: <https://cent.felk.cvut.cz/courses/PGR/lectures/08_Pipeline.pdf>. [Online; Datum: 2017-05-14].

[41] Petr Felkel. PGR - prezentace, 2017. Dostupné z: <https://cent.felk.cvut.cz/courses/PGR/lectures/08-zobrRetezec.pdf>. [Online; Datum: 2017-05-14].

[42] PROKOP, M. et al. Use of maximum intensity projections in CT angiography: a basic review. *Radiographics*. 1997, 17, 2, s. 433–451.

[43] RHEINGANS, P. – EBERT, D. Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics*. 2001, 7, 3, s. 253–264.

[44] ROTH, S. D. Ray casting for modeling solids. *Computer graphics and image processing*. 1982, 18, 2, s. 109–144.

[45] SABELLA, P. A Rendering Algorithm for Visualizing 3D Scalar Fields. *SIGGRAPH Comput. Graph.* June 1988, 22, 4, s. 51–58. ISSN 0097-8930. doi: 10.1145/378456.378476. Dostupné z: <http://doi.acm.org/10.1145/378456.378476>.

[46] SHIRLEY, P. – TUCHMAN, A. *A polygonal approximation to direct scalar volume rendering*. 24. San Diego, California, USA : ACM, 1990.

[47] The graphic pipeline. The graphic pipeline — Wikipedia, The Free Encyclopedia, 2015. Dostupné z: <http://en.wikipedia.org/wiki/Graphics_pipeline>. [Online; Datum: 2015-05-19].

[48] TREAVETT, S. M. – CHEN, M. Pen-and-ink rendering in volume visualisation. In *Proceedings of the conference on Visualization'00*, s. 203–210. IEEE Computer Society Press, 2000.

[49] UDUPA, J. K. – HERMAN, G. T. *3D imaging in medicine*. Philadelphia, PA, USA : CRC press, 1999.

[50] UPSON, C. – KEELER, M. V-buffer: Visible Volume Rendering. *SIGGRAPH Comput. Graph.* June 1988, 22, 4, s. 59–64. ISSN 0097-8930. doi: 10.1145/378456.378482. Dostupné z: <http://doi.acm.org/10.1145/378456.378482>.

[51] VIOLA, I. – GRÖLLER, E. Smart visibility in visualization. In *Computational Aesthetics*, s. 209–216, 2005.

[52] VIOLA, I. – KANITSAR, A. – GROLLER, M. E. Importance-Driven Volume Rendering. In *Proceedings of the Conference on Visualization '04*, VIS '04, s. 139–146, Washington, DC, USA, 2004. IEEE Computer Society. doi: 10.1109/VISUAL.2004.48. Dostupné z: <http://dx.doi.org/10.1109/VISUAL.2004.48>. ISBN 0-7803-8788-0.

[53] VIOLA, I. – KANITSAR, A. – GROLLER, M. E. Importance-driven feature enhancement in volume visualization. *IEEE Transactions on Visualization and Computer Graphics*. 2005, 11, 4, s. 408–418.

[54] WEISKOPF, D. *GPU-Based Interactive Visualization Techniques (Mathematics and Visualization)*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2006. ISBN 3540332626.

[55] ZDROJEWSKA, D. Real time rendering of heterogeneous fog based on the graphics hardware acceleration. *Proceedings of CESCG*. 2004, 4.

[56] ŠTĚTOVSKÁ, M. Ilustrace pohybu mechanických součástí. Diplomová práce, České vysoké učení technické v Praze, 2014.