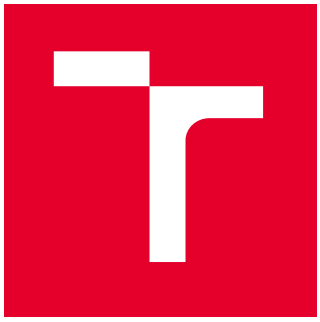
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2022 Michaela Viktorínová



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

DEPARTMENT OF BIOMEDICAL ENGINEERING

PROGRAM PRO ZOBRAZENÍ A PRÁCI S OBJEMOVÝMI DATY

SOFTWARE FOR VISUALIZATION AND PROCESSING OF VOLUMETRIC DATA

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Michaela Viktorínová

AUTHOR

VEDOUCÍ PRÁCE — Ing. Vratislav Harabiš, Ph.D.

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Biomedicínská technika a bioinformatika**

Ústav biomedicínského inženýrství

Studentka: Michaela Viktorínová ***ID:*** 220948

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Program pro zobrazení a práci s objemovými daty

POKYNY PRO VYPRACOVÁNÍ:

1) Proveďte literární rešerši základních přístupů pro zobrazování objemových dat. Zaměřte se především na metody pro přímé zobrazování objemů. 2) Proveďte rovněž rešerši metod zpracování vícerozměrných obrazů, které se používají pro přípravu dat k vizualizaci (segmentace, detekce, transformace dat, apod.). 3) Ve zvoleném programovacím jazyku navrhněte strukturu programu, který by měl sloužit pro zobrazení, manipulaci a základní práci s objemovými daty. 4) Navržený program implementujte. 5) Proveďte testování a hodnocení funkce programu, zaměřte se rovněž na výpočetní a paměťovou náročnost.

DOPORUČENÁ LITERATURA:

[1] ŽÁRA, J., BENEŠ, B., SOCHOR, J., FELKEL, P: Moderní počítačová grafika (2. vydání). Computer Press, 2005, ISBN 80-251-0454-0.

[2] UDUPA, J. K., HERMAN, G. T.: 3D imaging in medicine (2nd edition). CRC Press, 1991, ISBN:0-8493-4294-5.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 27.5.2022

Vedoucí práce: Ing. Vratislav Harabiš, Ph.D.

doc. Ing. Jana Kolářová, Ph.D.
předseda rady studijního programu

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Práce souhrnně popisuje možnosti zobrazení objemových dat. Blíže se věnuje skalárnímu typu objemových algoritmů, který následně dělí na techniky nepřímého a přímého zobrazení objemových dat. Z nepřímých technik přibližuje metodu Marching Cubes. Přímé techniky dále člení na algoritmy triviální a pokročilé. V praktické části popisuje návrh, implementaci a vyhodnocení funkčnosti programu, jenž využívá právě triviálních metod. Celkové hodnocení je složeno z dílčích hodnocení výpočetní náročnosti, zatížení paměti a kvality výsledného zobrazení za použití různých metod a druhů vstupních objemových dat.

KLÍČOVÁ SLOVA

Volumetrická data, algoritmy zobrazující objem, projekce maximální intenzity, projekce zprůměrované intenzity, projekce nejbližších cév, předzpracování, Python, Jupyter Notebook

ABSTRACT

The work summarizes the possibilities of volume data displaying. It deals with the scalar type of volume algorithms, which are then divided into techniques of indirect and direct rendering of volume data. From indirect techniques the Marching Cubes method is mentioned. Direct techniques are further divided into trivial and advanced algorithms. The practical part describes the design, implementation and evaluation of the functionality of implemented program that uses trivial methods. The overall evaluation consists of partial evaluations of the computational complexity, the memory load and the quality of final rendering while using different methods and types of input volume data.

KEYWORDS

Volumetric data, volume rendering algorithms, Maximum Intensity Projection, Average Intensity Projection, Closest Vessel Projection, pre-processing, Python, Jupyter Notebook

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

VIKTORÍNOVÁ, Michaela. *Program pro zobrazení a práci s objemovými daty*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství, 2022, 68 s. Bakalářská práce. Vedoucí práce: Ing. Vratislav Harabiš, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Michaela Viktorínová
VUT ID autora:	220948
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Program pro zobrazení a práci s objemovými daty

Prohlašuji, že svou závěrečnou práci jsem vypracovala samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autorky*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Tímto bych chtěla poděkovat vedoucímu své bakalářské práce, panu Ing. Vratislavu Harabišovi, Ph.D., za jeho odborné vedení, konzultace, trpělivost, vstřícnost a hlavně podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Bez možnosti vizualizace objemových dat si dnes již nemůžeme představit fungování nejednoho oboru. Na denní bázi ji využívají obory jako je např. fyzika, geografie, geologie, námořnictví, průmysl, biologie atd. V mé práci jsem se však zaměřila na využití zobrazení výše zmíněného typu dat v medicíně.

Samotný proces je nefotorealistickou vizualizační technikou. Cílem je zobrazení 2D projekce objemových dat, která si zjednodušeně můžeme představit jako 3D obrázky. Tato data jsou tedy trojrozměrná, diskrétní a v medicíně velmi často slouží k analýze údajů získaných z modalit jako je CT scan (Computed Tomography) nebo také MR (Magnetic Resonance Imaging).

Právě díky kombinaci prudkého nárůstu výkonu dnešní výpočetní techniky, dostupnosti nových technologií a neustálému nátlaku na rozvoj moderní medicíny se objemová vizualizace v posledních letech velmi rychle vyvíjí.

Mým cílem bylo v teoretické části této práce shrnout výsledky mnou provedené rešerše ohledně existujících základních přístupů pro zobrazení objemových dat, s převážným zaměřením na metody přímého zobrazování objemů, tzv. direct volume rendering. Dále také uvádím poznatky o metodách pre-processingu vícerozměrných obrazů, jenž jsou používány k přípravě zmíněných dat před samotnou vizualizací.

V praktické části jsem uplatnila provedenou řešerši z teoretické části práce při návrhnu struktury programu, jenž má za úkol zobrazit objemová data a zároveň umožňuje základní manipulaci s objemovými daty. Navržený program byl následně implementován, popsán a bylo na něm provedeno testování za účelem zhodnocení výpočetní a paměťové náročnosti a funkčnosti samotného algoritmu.

Výše zmíněné hodnocení bylo dále doplněno také o výstupy algoritmu a o poznatky založené na porovnání tří zobrazovacích metod, jenž byly v algoritmu užity, a to Maximum Intensity Projection (MIP), Average Intensity Projection (AIP) a Closest Vessel Projection (CVP).

1 Volumetrická data

Volumetrická data (také objemová data) si můžeme přiblížit na základě podobnosti s datovými prvky, ze kterých se skládají 2D obrázky. Každý pixel (anglicky „picture element“) uchovává informaci o hodnotě barvy, kterou reprezentuje. Takto definované pixely jsou uspořádány do mřížky (anglicky „grid“), která tvoří výsledný 2D obraz [9].

Abychom získali volumetrický objekt, musíme se přesunout z dvojrozměrného prostoru do trojrozměrného. Zmíněný volumetrický objekt si zjednodušeně můžeme představit jako sadu několika 2D řezů skládaných za sebe s definovanými rozestupy. Přidáním hodnoty rozestupů dostáváme třetí rozměr, z pixelů se tak analogicky stávají voxely (anglicky „volume element“). Z uvedených poznatků víme, že voxel je částicí objemu a je uložený v 3D mřížce. Voxel dále můžeme chápat nejen jako objemový element, jehož jedna hodnota popisuje celou kubickou buňku, ale také jako hranový bod této buňky. V takovém případě je bod uvnitř buňky definován interpolací [36].

Pokud se dále zaměříme na samotný 3D obraz, můžeme ho v počítačové grafice popsat pomocí rastrové nebo vektorové reprezentace.

1.1 Rastrová reprezentace

Rastrová reprezentace přistupuje k objektu jako k množině objemových elementů (voxelů), kde každý voxel je definován souřadnicemi na základě umístění v 3D mřížce. Existuje několik druhů mřížek. Každá uchovává své voxely odlišně.

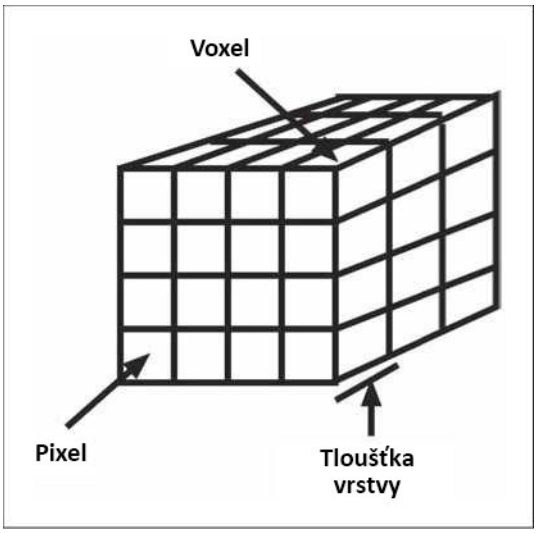
Rovnoběžná mřížka

Mřížka je představena jako matice o rozměrech $K \times L \times M$, kde jsou stěny jednotlivých buněk vzájemně rovnoběžné.

Dále v textu budu předpokládat, že vzorky dat jsou umístěny právě v pravidelné rovnoběžné mřížce obdobně jako je vyobrazeno na Obr. 1.1.

Mřížka s pravidelnou topologií

Opět zde mřížku reprezentujeme maticí $K \times L \times M$, ale přidáváme zde navíc síť parametrických ploch, což způsobí, že buňky v mřížce mají např. pouze stejný počet stěn.



Obr. 1.1: Pravidelná rovnoběžná mřížka. Dostupné z (upraveno): [37]

Mřížka s nepravidelnou topologií

Matice $K \times L \times M$ má zcela libovolně rozmístěné uzly, což způsobí vytvoření nepravidelné sítě buněk, ty mohou mít tvary např. čtyřstěnů (případně v rovině trojúhelníků).

Hybridní mřížka

Hybridní mřížka kombinuje předchozí pravidelnou a nepravidelnou topologii.

1.2 Vektorová reprezentace

Vektorový přístup popisuje 3D obraz pomocí grafických primitiv, jež jsou definována matematicky. Pro popis 3D objektů můžeme využít tři typů modelů. Použití každého typu záleží na geometrii vektorizovaného objektu [28]. Všeobecně vektorový přístup je využíván v oblastech, kde klademe důraz na přesnost.

Hranový model (Wire-frame)

Je popsán vrcholy a hranami, jež jsou spojnicemi vrcholů. Speciálním případem je využití křivek řezů, které reprezentují obrysy objektu.

Hranové modely jsou sice jednoduché na implementaci, nicméně nejsou příliš využívány, jelikož obsahují minimum informací a ne vždy je lze jednoznačně interpretovat [28].

Povrchový model (Boundary representation)

Povrchový model definuje hranice objektu na základě svého povrchu. K popisu hranic používá buď polygony, tvořené z bodů a křivek, nebo pro dosažení vyšší přesnosti aproximace povrchu se využívají komplexnější popisy za pomoci spline plochy (Beziérova plocha, B-spline, NURBS atd.) [28, 36].

Objemový model

Posledním typem je přístup, který pracuje také s vnitřními body objektu a popisuje tak i jeho objem. Model kombinuje navzájem objemová primitiva (hranoly, koule, válce atd.) za pomoci geometrických a množinových operací [28].

2 Možnosti vizualizace dat

Velikost dat, jež jsou pomocí zmíněných modalit naměřena, případně vytvořena simulacemi, se běžně pohybují v desítkách megabytů, někdy i v jednotkách gigabytů. Z toho plyne zřejmý požadavek na vysokou efektivitu zobrazovacích algoritmů [31].

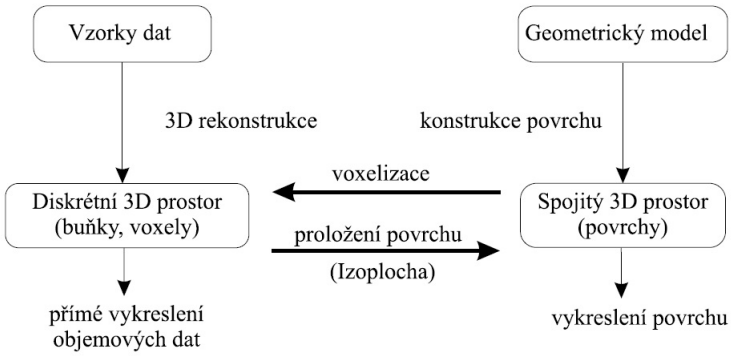
Dle typu zpracovávaných dat lze metody rozdělit do dvou skupin [31]:

- První skupina zobrazuje skalární plošné, prostorové či vícerozměrné mřížky nebo jinak uspořádaná skalární data.
- Druhá skupina pracuje naopak s vektorovými či tenzorovými poli, která následně zobrazuje. Tento typ dat obsahuje velké množství informací, navíc pole jsou vnitřně provázaná. Proto je třeba brát v úvahu, že je třeba pro zobrazení implementace speciálních algoritmů.

2.1 Skalární objemové algoritmy

V mé práci se budu blíže věnovat vizualizačním algoritmům, jež jsou určeny k zobrazení skalárních objemových dat (tedy první skupině zmíněné v úvodě kapitoly 2), neboť se jedná v praxi o nejběžněji používané algoritmy [36].

Algoritmy vizualizující skalární prostorové mřížky lze také rozdělit, a to na algoritmy zobrazující povrchy (anglicky „surface rendering“) a na algoritmy zobrazující objem (anglicky „volume rendering“). Srovnání metod znázorňuje Obr. 2.1.



Obr. 2.1: Postupy zobrazení objemových dat. Dostupné z: [36]

2.1.1 Algoritmy zobrazující povrchy

Algoritmy, zaměřující se na zobrazení povrchu, vytváří ze vstupních dat (mřížka skalárních hodnot – voxely) pomocnou geometrickou strukturu, jenž vyjadřuje povrch celého objektu. Je nutné zdůraznit, že surface rendering využíváme, pokud nás při reprezentaci dat nezajímá vnitřní stavba.

Tyto metody souhrnně označujeme jako nepřímo pracující, neboť před samotným zobrazením dojde k vytvoření pomocné povrchové masky, ta je reprezentována jednoduššími geometrickými tvary (nejčastěji trojúhelníky).

Pro převod vstupních dat do povrchové reprezentace se používají algoritmy mezi, které patří například Contour Connecting, Opaque Cubes, Dividing Cubes, Stretching Cubes [3]. Já se ve své práci více zmíním o metodě Marching Cubes.

Obr. 2.2: Povrchové zobrazení modelu kosti. Dostupné z: [31]

Marching Cubes

Algoritmus představili pánové Lorensen a Cline [16]. K popisu povrchu sítí používá síť trojúhelníků. Vrcholy těchto trojúhelníků se nachází na hranách jednotlivých buněk (voxelů). Buňky jsou postupně označeny binárním systémem buď za zasažené (1) či nezasažené (0). Klasifikace probíhá na základě hustoty voxelů, jenž porovnáváme s definovanou hranicí. Metoda zná celkem 256 případů, kdy je buňka protnuta trojúhelníky, počet se však dá redukovat na 15 základních konfigurací pomocí rotace nebo symetrií (Obr. 2.3) [36].

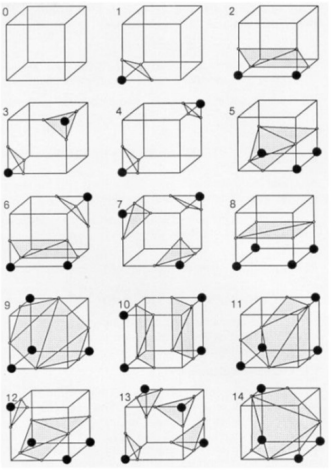
Marching Cubes takto vytvoří prostorovou masku složenou z množiny 2D polygonů, která vymezuje části prostorových dat, jež budou dále zobrazeny, případně jinak zpracovány.

Jedná se o rozsáhlé a výpočetně náročné vykreslování, které není vhodné pro renderování v reálném čase [33].

Rozšířením Marching Cubes je metoda Marching Tetrahedra publikovaný v [27].

2.1.2 Algoritmy zobrazující objemy

Tyto metody zobrazují skalární data přímo, mluvíme tedy o tzv. přímém volume renderingu. Na rozdíl od nepřímého volume renderingu nemusí mít definované, zdali



Obr. 2.3: 15 základních konfigurací v algoritmu Marching Cubes. Dostupné z: [16]

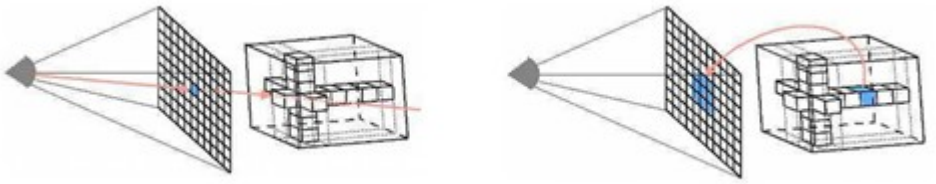
voxel patří či nepatří k zobrazovanému povrchu, jelikož nevytváří pomocnou geometrickou síť.

Hlavním rozdílem od surface renderingu je přímé získání 2D obrazu z objemových dat. Extrahujeme informace z 3D pole skalárních hodnot, čímž mapujeme vizuální vlastnosti dat (barva, neprůhlednost). Celý proces mapování obstarává přechodová funkce. Příkladem takové funkce je přímé přiřazení barvy konkrétní hodnotě objemové veličiny. Díky variacím přechodových funkcí je proces mapování velice flexibilní. Proto klademe na přechodové funkce velký důraz.

Algoritmy se uplatní tam, kde informace získané prostým povrchovým zobrazením nejsou dostatečné, umožňují totiž současně vizualizovat rozhraní mezi strukturami a jejich vnitřní strukturu.

Rychlost vykreslování je u metod přímého volume renderingu velmi odlišná. Záleží na komplexnosti užité metody. Triviální algoritmy patří mezi rychleji renderující metody, avšak kvalita jejich výstupu je mnohem horší než u složitějších metod. Všeobecně musíme u přímého volume renderingu počítat s tím, že všechny informace obsažené v datech mohou být stěžejní ve výsledném obraze, což nám zkomplikuje výsledné zobrazení. Zároveň při každém otočení scény musíme myslet na to, že obraz bude přepočítán a znovu renderován [34]. Metody, jenž nabízejí rozdílné kombinace rychlosti zobrazení a kvality výsledné scény, jsou uvedeny v následujících publikacích [5, 14].

I přímý volume rendering můžeme dále rozdělit do dvou kategorií s rozdílnými přístupy k zobrazení dat. Mluvíme o tzv. Image-order algoritmech a object-order algoritmech. Přístupy se převážně liší v rychlosti zobrazení, jelikož Image-order pracuje ve 2D prostoru, kde data zpracovává po jednotlivých pixelech. Naopak object-order přístup využívá 3D prostor a zpracovaná objemová data jsou následně promítána [9].



Obr. 2.4: Vykreslení dat pomocí Image-order (vlevo) a Object-order technik (vpravo). Dostupné z: [9]

2.1.3 Triviální metody DVR

Maximum Intensity Projection (MIP)

MIP je velmi rychlý algoritmus, jenž je velmi často využíván například právě v medicínském prostředí (zobrazení jasných cév při CT angiografii) [24, 7]. Metoda využívá zjednodušenou podobu ray-tracing techniky (česky „sledování paprsku“), která je v tomto případě založena na vykreslování maximální intenzity. Vržený paprsek postupně prochází objemovými daty a na projekční rovinu (obraz) zobrazuje nejjasnější strukturu (maximální intenzitu) ze všech měřených intenzit buněk v rámci vyslaného paprsku. Tato maximální hodnota poté definuje výslednou barvu (intenzitu) zobrazovaného pixelu [36, 40]. Proces shrnuje matematický vztah:

$$I = \max_{i \in J} (I_i) \tag{2.1}$$

kde I představuje vypočítanou intenzitu jasu pixelu v zobrazovaném obraze, I_i je skalární hodnota intenzity naměřená podél paprsku a J označuje množinu vzorků zasažených vyslaným paprskem.

Při zobrazování pomocí MIP dochází k ztrátě objemové informace, proto je žádoucí renderovat záběry z různých úhlů pohledu.

Closest Vessel Projection (CVP)

Metoda v odborné literatuře také často označovaná jako Local Maximum Intensity Projection (LMIP) je modifikací výše představené MIP metody.

Closest Vessel Projection stejně jako MIP vysílá paprsek, ten prochází objemovými daty, tentokrát paprsek ale nehledá maximalní hodnotu podél celé své délky, nýbrž první lokální maximální hodnotu, která je větší než předdefinovaná prahová hodnota.

Využití kombinace prahování a lokálního maxima umožňuje zobrazovat na projekci i méně výrazné cévy, jenž se vyskytují v popředí vyslaného paprsku [25]. Při

zobrazování metodou MIP by mohly být zmíněné cévy přehlédnuty v případě, že by se dále v cestě paprsku vyskytovala více jasnější struktura (větší céva, kost).

CVP je často využívaná k vizualizaci renálních cév při CT angiografii či k zobrazení mozkových cév, jež jsou v blízkosti aneurysma, při MR angiografii [25].

Matematický vztah 2.1 lze upravit do podoby:

$$I = \max_{k \in J} (I_k) \tag{2.2}$$

kde je předpokladem existence ryzího okolí $\bar{O}(k)$ v rámci množiny paprskem zasažených vzorků J. Pro ryzí okolí musí platit $I_k \geq I_i$ a to pro všechna $i \in \bar{O}(k)$ [18].

Summed Intensity Projection

Součtová metoda je alternativní metodou MIP, pomocí které můžeme vypočítat hodnotu intenzity pixelu. Výsledná hodnota je rovna sumě všech intenzit podél vrženého paprsku [4, 36]. Matematicky tuto operaci popisuje vztah:

$$I = \sum_{i \in J} (I_i) \tag{2.3}$$

kde význam značení je totožný jako v metodě MIP.

Average Intensity Projection (AIP)

Průměrovací metoda je velmi podobná metodě MIP. Avšak liší se ve výsledném výpočtu hodnoty intenzity zobrazovaného pixelu. Zde je výsledek vypočítán jako průměr všech intenzit podél paprsku, tudíž dochází k normalizování příspěvků různého počtu stejných intenzit [4, 36]. Operaci zapíšeme následovně:

$$I = \frac{1}{|J|} \sum_{i \in J} (I_i) \tag{2.4}$$

význam značení je opět totožný jako v metodě MIP.

Obrazy na výstupu této metody mají zpravidla menší množství šumu a jejich hrany jsou hladší. Proto je AIP využíváno například při zobrazování vnitřních struktur některých orgánů nebo stěn dutých struktur jako jsou cévy nebo střeva [4].

2.1.4 Pokročilé metody DVR

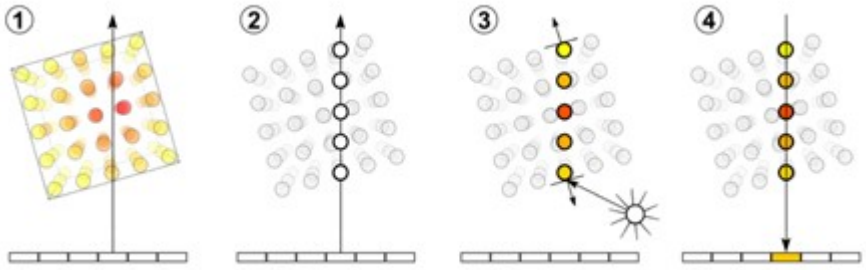
Ray Tracing/Casting

Nejvíce používaná Image-order technika pro vykreslování volumetrických dat je Ray Tracing (RT) [9, 33]. Metoda přímo zobrazuje, co reálně vidíme z kamery. Zároveň se snaží o simulaci fyzikálních vlastností a chování světla.

RT sleduje trasu paprsku ve směru od kamery přes scénu ke světelnému zdroji, proto se metoda také jmenuje metoda „zpětného sledování paprsku“. Vysílané paprsky vnímáme jako samostatné nezávislé procesy. Paprsky vysíláme ze středního bodu projekce přes rovinu výsledného obrazu do samotné scény. Podél každého paprsku se v pravidelných intervalech akumuluje hodnota barvy (dáno strukturou prostředí objektu a barvou pozadí) získaná přenosovou funkcí. V paměti se tak udržuje hodnota z posledního kroku trasování, ke které se přidávají další hodnoty získané z aktuálních vzorků. Na výstupu tak máme hodnotu RGB barvy (můžeme na konci upravit, např. stínováním) a neprůhlednosti jednoho pixelu výsledného obrazu, kterému vyslaný paprsek odpovídal [9, 33, 13].

Často však při vykreslování objemových dat nepotřebujeme celý RT, proto využíváme spíše metodu Ray Casting (RC, česky „vrhání paprsků“). RC opět sleduje průchod paprsku prostředím, nicméně průchod není ovlivněn odrazem a lomem jako u RT. U RC známe objem zobrazovaného objektu, proto do akumulátoru přičítáme hodnoty podél paprsku skrze celý objem a ne hodnoty vypočítané odrazem či lomem. Efekt lomu/odrazu by v medicínském prostředí měl minimální užitek [33].

Proces je pomalý, jelikož k výpočtu jednoho pixelu musí projít jeden paprsek celým objemem scény, proto byly vyvinuty různé způsoby pro urychlení procesu, a to například Early Ray Termination či Space Leaping [9].



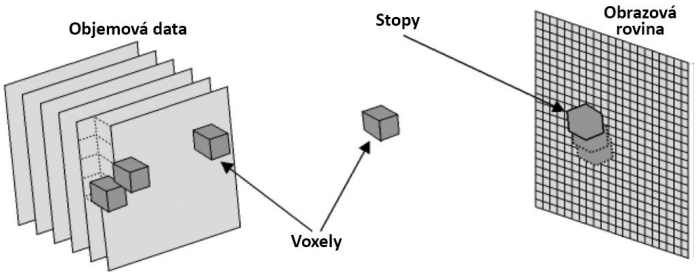
Obr. 2.5: 1. vyslání paprsku pro daný pixel scény, 2. vzorkování, 3. interpolování hodnot a použití transfer funkce, 4. výpočet integrálu pro zisk RGB hodnoty pixelu. Dostupné z: [9]

Voxel Splatting

Ze zástupců kategorie Object-order zmíním metodu Voxel Splatting (česky „plácání“). Jedná se o algoritmus, který hlavně klade důraz na rychlost procesu zobrazení a to i za cenu nižší kvality svého výstupu (obraz může být rozmazaný) [9].

Metoda hledá pro každý voxel 3D dat pixely výsledného obrazu, u kterých daný voxel ovlivní jejich hodnotu. Postupně tak na sebe skládá („plácá“) jednotlivé voxely

na zobrazovanou rovinu, nejdříve od nejzvdálenějších voxelů 3D obrazu až k těm nejbližším. Příspěvek jednotlivých voxelů se hromadí, až dojde k vytvoření finální 2D projekce, které se říká Footprints (česky „stopy“). Stopy mají různě definované parametry (velikost, barva, tvar atd.), čímž dotvoří projekci výstupního obrazu [13, 36].



Obr. 2.6: Metoda Voxel Splatting. Dostupné z (upraveno): [13]

Shear Warp

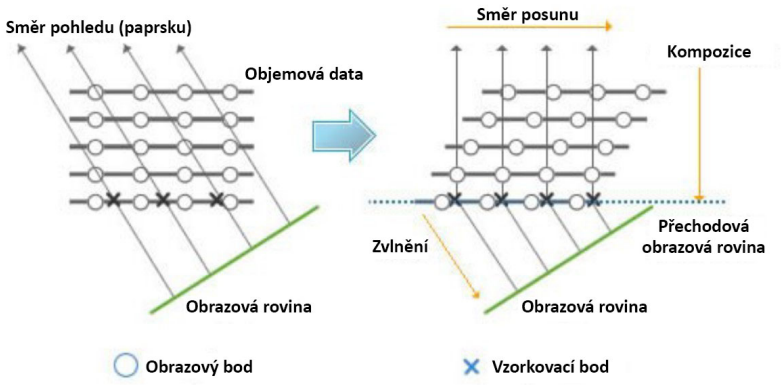
Shear Warp (česky „posun a zvlnění“) můžeme zařadit mezi hybridní algoritmy, jelikož kombinuje výhody jiných přístupů (převážně Ray Casting) s inovativním přístupem tak, že je daleko méně časově a výpočetně náročný. Na druhé straně ovšem vytváří v konečné projekci artefakty, navíc samotná kvalita projekce je nízká [13, 9].

Objemová data nejdříve promítáme na dočasně vytvořenou obrazovou rovinu (projekci), kterou následně 2D transformací zdeformujeme na výsledný obraz.

Na začátku tedy zavádíme pomocný souřadnicový systém, který dle potřeby může být „zkosený“. V tomto systému se nachází transformované voxely. Na základě úhlu pohledu na objemová data definujeme, s kolikátou souřadnou osou je dočasná obrazová rovina rovnoběžná tak, aby její pixely byly zarovnány s voxely. Dojde tak k posunutí jednotlivých řezů objektu (Shear část metody). Právě rovnoběžnost, zarovnání a posunutí nám zajistí efektivní promítnutí na dočasnou obrazovou rovinu. Dále dochází k vzorkování objemových dat paprskem v rámci rovnoběžné mřížky a následnému promítnutí na dočasnou obrazovou rovinu. Na závěr musíme zdeformovaný obraz na dočasné obrazové rovině upravit zpět (Warp část metody), abychom získali konečnou projekci [15, 36].

Texture Mapping

Metoda Texture Mapping se snaží využít grafický hardware k urychlení procesu volume renderingu. Opět zde pracujeme s paprskem, který směřuje ze zdroje přes průmětnu do objemu. Podél tohoto paprsku vzorkujeme objemová data, které chceme renderovat. Pokud vzorky leží v rámci jedné roviny, můžeme ji nahradit polygonem



Obr. 2.7: Metoda Shear Warp. Dostupné z (upraveno): [13]

a na něm uskutečnit proces vzorkování. Poté už jen zbývá uložit objemová data jako 3D texturu.

Princip metody je tedy následující, vygenerujeme sadu polygonů v definovaných místech uvnitř objemového prostoru. Tyto polygony vzniknou ořezem rovnoběžných rovin s průmětnou pomocí vytvořené obálky z objemu. Na polygony poté namapujeme předvytvořenou 3D texturu a provedeme jejich navzorkování a interpolaci [13].

Výhodou algoritmu je snadná implementace a rychlý proces renderingu. Naopak nevýhodou je, že velikost potřebné paměti k provedení zobrazení rapidně roste s přibývajícím množstvím dat. Navíc vyrenderované obrazy mohou být silně postižené vizuálními artefakty [9].

3 Zpracování dat

Obecně platí, že získaný obraz obsahuje nejrůznější typy artefaktů a zkreslení, jenž samozřejmě zhoršují jeho kvalitu. Pre-processing obrazu by správně měl tyto negativní vlivy (šum, geometrické zkreslení, jasové zkreslení, rozostření atd.) co nejvíce potlačit a připravit snímek pro jeho následnou segmentaci a zobrazení.

Mezi běžně užívané pre-processingové postupy řadíme:

- filtrace šumu (průměrování, mediánová filtrace, Gaussovská filtrace, prahování vlnkových koeficientů, metody založené na parciálních diferenciálních rovnicích atd.)
- transformace jasové stupnice (ekvalizace histogramu, úprava kontrastu atd.)
- geometrické transformace (souhrnně zmíněno dále v části 3.2)
- ostření obrazu (ostřící maska)

Zmíněným typům filtrací se např. věnují následující literatury [12, 10].

Působení transformace jasové stupnice či filtrů můžeme rozdělit na základě velikosti okolí bodu, ze kterého působí. Jedná se o **globální**, **lokální** či **bodové** působení.

Po vhodně provedeném pre-processingu můžeme následně obraz segmentovat. Zde je důležité zdůraznit, že výsledek segmentace je mnohdy velmi závislý na kvalitně provedeném předzpracování dat. Jako příklad špatného předzpracování nebo také žádného předzpracování můžeme uvést nadbytečnou klasifikaci neexistujících objektů v místech, kde je původní obraz zatížen šumem.

K pre-processingu se velmi často také řadí detekce hran objektu, ta je zpravidla obstarávána metodou prahování či přímo hranovými detektory. K této problematice se blíže dostanu v části 3.1.

3.1 Segmentace

Pokud pracujeme s medicínskými daty, musíme brát v úvahu nejen pre-processing dat a následnou vizualizaci, ale častokrát musíme zahrnout i proces segmentace dat.

Lékaři musí být schopní vizuálně rozlišit konkrétní objekty, jež jsou předmětem zájmu v zobrazeném souboru dat, které z výše zmíněných modalit získali. Pokud tak učiníme, můžeme objekty nejen rozlišit (např. barevně), ale také i zanedbat při vykreslení zbytku dat.

Dnes je potřebná například ke kvantifikaci objemu dat, správnému detekování patologií, naplánování průběhu operací, či k samotnému navigování během chirurgické intervence [26, 29].

Jde o označení jednotlivých voxelů, tím vytváříme informaci o jejich příslušnosti k některému z konkrétních objektů (anglicky „object membership information“). Tento proces rozdělí data na homogenní oblasti (segmenty) podle předem definovaných pravidel a vlastností [9, 10].

3.1.1 Metody segmentace

Metod k provedení segmentace je v současné době velké množství, proto je třeba si nejdříve řádně promyslet, jaké máme požadavky na výstup, jaké jsou naše vstupní podmínky a jaké výhody a potažmo i nevýhody má užití daného přístupu. Je třeba si uvědomit, že neexistuje ideální metoda, která by nejlépe segmentovala daný druh tkáně, proto se pro dosažení nejlepšího výsledku často jednotlivé přístupy kombinují.

Metody si můžeme rozdělit do několika podskupin na základě různých kritérií [29, 10].

Pokud bereme v úvahu míru interakce s uživatelem: O **automatické** segmentaci mluvíme, pokud je obraz segmentován algoritmem samostatně bez zásahu uživatele, efektivita algoritmu závisí na nastavených parametrech, které si zvolený algoritmus stanoví sám. Při **poloautomatické** segmentaci se prolínají výhody manuálního a automatického přístupu. Segmentace je stále prováděna automaticky algoritmem, nicméně je zde míra interaktivity s uživatelem, který koriguje nastavené parametry. Výsledkem je zvětšení přesnosti procesu při relativním zachování časové nenáročnosti. U **manuální** segmentace dochází k označování struktur ručně uživatelem. Jelikož dochází k analýze samotným uživatelem, předpokládají se u tohoto přístupu hlubší znalosti anatomie uživatele a zároveň delší doba provedení analýzy. Je zde větší náchylnost k chybám, dále je to metoda subjektivní a nereprodukovatelná. Proto se v dnešní době dává přednost spíše předchozím dvěma metodám.

Dále můžeme proces segmentace dělit podle typu zpracovávaných dat: U **2D segmentace** vstupní data tvoří dvourozměrné snímky. **3D segmentace** naopak pracuje s trojrozměrnými daty (či mnoha dvourozměrnými daty zároveň). Je velice důležité brát v úvahu počet dimenzí a druhy parametrů, se kterými daná metoda pracuje, protože může dojít k odlišným výsledkům různých metod se stejnými vstupními daty. Za jistých okolností navíc můžeme aplikovat i 2D algoritmus na objemová data, čímž zjednodušíme implementaci, výpočetní náročnost nebo dokonce zlepšíme výsledek segmentovaných objektů.

3.1.2 Statické metody

Prahování

Jeden z nejstarších a nejsnadněji implementovatelných algoritmů. Základní myšlenkou je práce s histogramem, ten nese informaci o distribuci konkrétních hodnot pixelů v obraze. Metoda předpokládá, že snímané objekty vykazují rozdílnou intenzitu jasu oproti pozadí obrazu [20]. Cílem je stanovit hodnotu hraniční intenzity jasu (práh), která obraz segmentuje do skupin. Po definování prahu nastává roztřídění pixelů na ty, které mají jas vyšší nebo rovný prahu a nabudou zastupující hodnoty 1 a ty jenž mají jas nižší než definovaný práh a jejich zastupující hodnota bude 0. Tím se vytvoří binární obraz, kde definujeme oblasti zájmu od pozadí obrazu [29]. Transformaci do binární reprezentace můžeme zapsat také matematicky:

$$g(i,j) = \begin{cases} 1 \;\; pro\; f(i,j) \geq T \\ 0 \;\; pro\; f(i,j) < T \end{cases} \tag{3.1}$$

kde g je výstupní prahovaný obraz, f reprezentuje vstupní data a T je definovaný práh.

Pro co nejpřesnější výstup je stěžejní správná volba prahu T. Ten můžeme stanovit ve většině případů po důkladné analýze rozložení histogramu obrazu. Nejjednodušší variantou je stanovení lokálního minima (sedla) mezi dvěma maximy histogramu, jelikož právě zde předpokládáme mez oddělující tkáně. Histogram je velmi citlivý na šum a další artefakty, navíc pokud se v něm nevyskytuje výrazné sedlo, může nastat problém při určování prahu [22].

I přes nevýhody nevyhnutelně doprovázející popisovanou metodu, se tyto algoritmy hojně používají u segmentace kostí, kde je vyšší kontrast oproti zbytku obrazu [23].

3.1.3 Oblastně orientované techniky

Oblastně orientované techniky segmentace detekují oblasti v obraze na základě jejich homogenity. Homogenita je hlavním segmentačním kritériem a je definována parametry jako např. úrovní šedi, barvou, texturou, tvarem, modelem atd. [30]

Popsané metody vytvářejí segmenty přímo, nehledají totiž v obraze hrany, protože jsou zaměřené na vlastnosti samotných obrazů. To může být výhodné především v případě obrazů postižených šumem [12]. Dochází tak ke sdružování sousedních pixelů s podobnými vlastnostmi do tříd.

Narůstání oblastí

Myšlenkou této metody je seskupování homogenních pixelů postupně v několika iteracích. Nejdříve je třeba určit počáteční jádrový pixel (popř. pixely). Ten bude definovat vlastnosti segmentu, jenž se bude na základě rozhodovacího kritéria případně rozšiřovat. Vlastnosti jsou reprezentovány parametrem p, porovnání vyjadřuje následující definice [12]:

$$|p_s - p_j| \leq T \tag{3.2}$$

kde p_s je referenční parametr vzorového pixelu, p_j je parametr testovaného pixelu a T je rozhodovací kritérium.

Pokud pixely splňují kritérium T jsou začleněny do segmentované oblasti, v opačném případě jsou vynechány. Metoda pomocí popsaného rozhodovacího pravidla projde celý obraz. Rozhodovací kritérium může fungovat staticky či dynamicky. U statického případu se testovaný parametr pixelu porovnává s počátečním jádrovým pixelem. V druhém případě se testovaný parametr pixelu porovnává např. s naposledy přidaným pixelem. Proces je ukončen v momentě, když už není možné přiřadit další nový pixel nebo je překročeno možné množství iterací [29].

Rozdělení a slučování oblastí

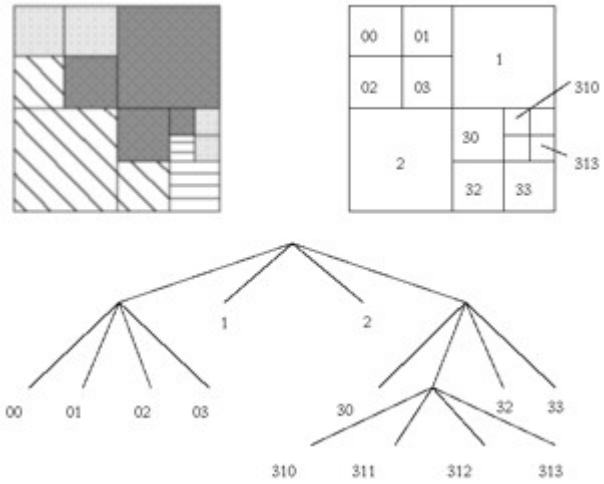
Metoda dělící a následně slučující oblasti funguje na principu dělení obrazu a následného spojování homogenních podskupin. Je to automatický proces, jenž probíhá opět iteračně.

Algoritmus funguje na tzv. quad-tree prezentaci dat (Obr. 3.1) [30]. V obraze je zkoumána homogenita na základě různých atributů (průměr, rozptyl, minimální a maximální hodnoty). Pokud není splněno kritérium homogenity, je obraz rozdělen na předem definovaný počet podobrazů, zpravidla na čtyři kvadranty (proto označení „quad“). Pokud nejsou homogenní ani dílčí kvadranty, proces se opakuje tak dlouho, dokud není kritérium splněno. Nakonec dojde ke spojení oblastí s dostatečně podobnými atributy [29].

Nevýhodou tohoto přístupu je vznik pravoúhlých segmentovaných oblastí, z čehož vyplývá nepřesnost v kopírování tvaru segmentovaných objektů [23, 26]. Naopak výhodou je odolnost vůči šumu a také zčásti přerušeným hranám snímaných objektů [12, 10].

3.1.4 Detekce hran

Jedná se o jednu z nejdůležitějších oblastí nižší úrovně zpracování obrazu. Hranou rozumíme oblast obrazu, kde se intenzita jasu prudce změní, hrana je tedy vyjádřena



Obr. 3.1: Princip hierarchie segmentace rozdělení a slučování oblastí. Dostupné z: [30]

velikostí a směrem [30].

Abychom ve snímku nalezli hrany, musíme provést výpočet gradientu. Hodnota gradientu definuje nejen změnu intenzity jasu, ale také i její směr. Alternativním přístupem k výpočtu hodnoty gradientu je konvoluce snímku se souborem vzorů hran. Na daném principu fungují hranové detektory.

Pomocí funkcí si můžeme přiblížit několik druhů hran. Skoková funkce (anglicky „step function“) vyjadřuje model ideální hrany. Nasnímaná data nikdy nemohou být zcela ideální, a tak změna intenzity jasu nikdy nebude skoková, nicméně k ní bude docházet postupně, k popisu postupné změny proto používáme šikmou funkci (anglicky „ramp function“). Může nastat situace, kdy výše zmíněné funkce budou na snímku vystupovat vedle sebe. V takovém případě vznikají další typy hran, např. čára (anglicky „line function“) či střecha (anglicky „roof function“). Popsané hrany shrnuje Obr. 3.2 [30]:

Obr. 3.2: Typy hran v obraze. Zleva: step, ramp, line a roof. Dostupné z: [30]

Detekce hran probíhá v podstatě ve čtyřech krocích. Nejdříve je třeba obraz filtrovat. Dá se předpokládat, že dojde ke vzniku šumu a artefaktů např. vzorkováním, kvantováním, případně rozmazáním nebo také nevhodným nastavením snímací techniky. Správně zvoleným filtrem můžeme alespoň částečně tyto parazitické jevy odstranit, zvláště pak kvůli odstranění šumu filtrujeme vyšší frekvence. Následuje diferenciace, která zvýrazní hrany a to díky výraznějším změnám v intenzitě jasu

snímku. Poté dojde k detekci bodů, kde jsou změny jasu nejvýraznější. Nakonec jsou detekované body pospojovány do řetězců tak, aby výsledné spojení odpovídalo se skutečnými objekty obrazu.

Hranové detektory

Hranové detektory využívají k analýze hran konvoluci obrazu s konvoluční maskou, její koeficienty a rozměry reprezentuje typ hranového operátoru (např. Cannyho, Laplacův, Prewittové, Sobelův, Robertsův). Podle principu fungování rozdělujeme detektory na několik druhů, za zmínku zcela jistě stojí detektory hledající maxima prvních derivací obrazové funkce, detektory průchodu nulou u druhých derivací obrazové funkce nebo detektory založené na aproximaci obrazové funkce polynomem [12, 10, 30].

Hranová detekce první derivací

U diskrétních obrazů získáme jejich první derivaci rozdílem okolních pixelů obrazu. Derivaci je možné v tom nejlehčím možném případě vypočítat zvlášť pro řádky (respektive sloupce). Gradient je poté vypočítán podle vzorce:

$$G\left(i,j\right) = \sqrt{G_R\left(i,j\right)^2 + G_S\left(i,j\right)^2} \tag{3.3}$$

kde sousední pixely jsou dosazovány postupně zleva doprava, respektive shora dolů. Pro určení orientace gradientu v obraze slouží následující vztah (daný tvar popisuje orientaci vzhledem k řádkům):

$$\theta\left(i,j\right) = \tan^{-1}\frac{G_S\left(i,j\right)}{G_R\left(i,j\right)} \tag{3.4}$$

Zisk hodnoty gradientu si můžeme představit jako konvoluční filtrování signálu s n-dimenzemi. Jak již bylo nastíněno v 3.1.4 detektory se mezi sebou odlišují jádrem v konvolučním filtru. Tato jádra stanoví body užívané pro výpočet gradientu a velikost jejich váhy. Z toho je zřejmé, že velikost a nastavené hodnoty konvolučního filtru přímo ovlivňují výsledné vlastnosti hranového detektoru. Výsledný obraz po užití konvolučního filtru se nazývá gradientní, jedná se o výsledek konvoluce originálního snímku a jádra filtru [30].

Samotná detekce první derivací využívá prahování. Hraniční hodnota stanoví zda je hodnota gradientu dostatečně významná. Tento proces zajišťují právě již popsaná jádra konvolučních filtrů. Jelikož jádra realizují detekci pouze v jednom směru, často se využívají jádra v párech, tak aby jedno jádro detekovalo hrany ve vertikálním směru a druhé v horizontálním [30].

Cannyho hranový detektor umožňuje vyhledávat různé varianty rozlišení, z nich následně vyhodnotí to nejvhodnější a realizuje ho [22].

Výsledný výstup detektoru je prahován, čímž získáme informaci o významných a nevýznamných hranách. Jsou určeny dva prahy $T_1 < T_2$. Pro jednotlivé pixely obrazové funkce $f(x, y)$ tak platí:

$$(x, y) = \begin{cases} f(x, y) \geq T_2 & (x, y) \text{ je hranový bod;} \\ T_1 \leq f(x, y) < T_2 & (x, y) \text{ je hranový bod pokud } \exists \text{ sousední hran. bod;} \\ T_1 \leq f(x, y) < T_2 & (x, y) \text{ není hranový bod pokud } \nexists \text{ sousední hran. bod;} \\ f(x, y) < T_1 & (x, y) \text{ není hranový bod.} \end{cases} \tag{3.5}$$

Houghova transformace

Jedná se o speciální případ hranových detektorů, kdy pomocí transformace se snažíme vytvořit parametrický model hran z funkce obrazu. Dochází zde k transformaci mezi souřadnicovými systémy, jelikož data přechází z kartézského systému do polárního [30]. Často je tato metoda uplatňovaná k zpřesnění hrubé hranové reprezentace. Nicméně podmínkou užití je, že se v ní musí nacházet určité definované tvary (např. přímka, kruh, elipsa atd.), u kterých ale mohou existovat jisté nepřesnosti nebo jsou také přerušené či zkreslené.

Množství parametrů, jenž popisují onen tvar v obraze definuje počet dimenzí parametrického prostoru, do kterého přenáší transformace hranovou reprezentaci obrazu.

Výhodou této metody je bezesporu přesnější popis hrany, navíc celý proces je velmi robustní. Avšak jedná se o výpočetně velmi náročnou operaci.

Houghova transformace je nevhodná k segmentaci např. mozku, poškozené tkáně, jelikož jeho struktury neobsahují pevně definované tvary [12, 10].

Nejčastěji se s touto metodou setkáme u detekování přímek v obraze. Ty popisuje následný matematický vztah:

$$y = m \cdot x + b \tag{3.6}$$

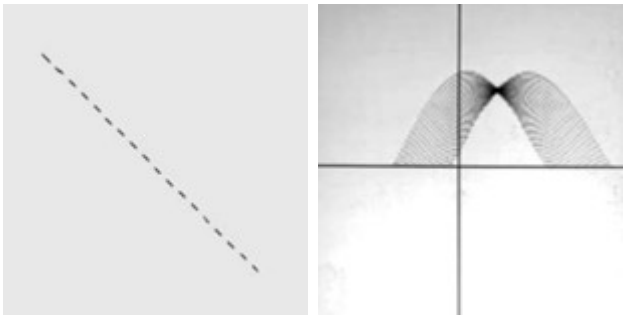
kde x, y jsou souřadnice bodů, kterými přímka prochází. A m, b jsou koeficienty určující sklon a posunutí přímky. Rovnice ale nevyhovuje svisle orientovaným přímkám, proto byl zavedený další vztah [20]:

$$d = x \cdot \cos\theta - y \cdot \sin\theta \tag{3.7}$$

kde d udává normalizovanou vzdálenost přímky od počátku souřadnic a θ je úhel svíraný přímkou vzhledem k horizontální ose. V tomto momentě v podstatě došlo

k transformaci do Houghova prostoru. Pokud budeme v takovém případě hledat přímku v obraze, vstupními daty budou souřadnice pixelů x a y a neznámými parametry d a θ. Pokud do rovnice dosadíme vstupní data, řešením bude množina řešení, které se v Houghově prostoru reprezentuje jako křivka. Pokud provedeme dosazení i pro další souřadnice pixelů tvořících v obraze přímku, dojde v Houghově prostoru k protnutí křivek v bodě, jenž je definovaný parametry d a θ. Bod protnutí křivek tak popisuje hledanou přímku pomocí zmíněných parametrů d a θ a body původního obrazu jsou ve výsledku reprezentovány křivkami v parametrickém prostoru [30].

Proces Houghovy tranformace přímky je zachycený pomocí Obr. 3.3



Obr. 3.3: Houghova transformace přímky. Dostupné z: [30]

3.2 Prostorová transformace

V rámci 3D prostoru můžeme provádět různé typy transformačních operací. Každá transformační operace přepočítá souřadnice bodů dat z původní soustavy na souřadnice nové soustavy.

K popisu transformací zavádíme tzv. Degrees of Freedom (DOF, česky „počet stupňů volnosti“), což si můžeme představit jako počet potřebných parametrů. Číslo DOF závisí na dimenzi obrazů a na strukturách zobrazených v obrazech.

Transformace si můžeme rozdělit například podle míry složitosti dle DOF. Je zřejmé, že s rostoucí složitostí transformace, poroste také DOF, což by zpravidla mělo mít pozitivní vliv na správnost výsledků. Při provádění těchto operací je však třeba mít na mysli, že uvedenými transformacemi můžeme přijít o důležité informace, které se mohou hodit k různým typům analýz [8].

Rigidní transformace

Jedná se o nejjednodušší typ transformačních operací. Popisuje posun a rotaci objektu. V 3D prostoru je definována šesti DOF, což znamená, že bude potřebovat

k popisu šest parametrů, jedná se o parametry translace v osách x, y a z a rotace kolem zmíněných os [28].

Afinní transformace

Tento typ užíváme, pokud je mezi transformovanými soustavami rozdíl nejen v pozici, jelikož kromě translace, rotace definují dále i změnu měřítka a zkosení objektu. To znamená, že počet DOF musí vzrůst, pokud opět uvedeme příklad s 3D objektem, parametrů bude tentokrát dvanáct [28].

Nelineární transformace

Nelineární transformace jsou nejsložitější transformační operace. Popisují jak globální změny objektu (translace, rotace, zkosení), tak určují i změny na lokální úrovni. Mezi takové transformace můžeme řadit např. lokální deformace [28].

4 Jupyter Notebook

V rámci praktické části jsem se rozhodla realizovat mnou navržený algoritmus v programovacím jazyce Python. Nástroj pomocí jehož jsem algoritmus vytvářela se nazývá Jupyter Notebook.

Jupyter Notebook je nástroj, který nabízí přehledné a interaktivní prostředí využívající webové rozhraní. Jupyter Notebook vychází z projektu IPython Notebook(s), a tak není překvapením, že je primárně orientován na uživatele používající programovací jazyk Python [32].

4.1 Využití

Jupyter Notebook pracuje na principu REPL (Real-Eval-Print-Loop), což umožňuje prakticky ihned vyhodnocovat uživatelem zapsané požadavky. To se může hodit v případě, kdy potřebujeme výsledky zobrazit okamžitě nebo když chceme např. postupovat krok po kroku. Mimo výrazy dokáže tento nástroj také pracovat s obrázky, grafy či tabulkami, dokonce si poradí i s matematickými vzorci z TeXu a LaTexu, tvorbou slidů či živého zdrojového kódu. Rozhraní tak může připomínat obdobu diáře (proto název notebook), kde je možné zapsat nejen samotný kód, nechat vypsat výsledky daného kódu, ale zároveň si pomocí funkce Markdown přehledně zaznačit ke všemu poznámky (obdobnou technologii nalezneme například i v Matlabu) 4.

4.2 Technologie

Jak již jsem zmínila v úvodu kapitoly 4, Jupyter Notebook využívá webového rozhraní, tudíž oblíbené technologie klient-server. Klientem je v tomto případě spuštěný webový prohlížeč a serverem je Jupyter s tzv. kernelem (modul pro námi využívaný programovací jazyk). I když je Jupyter Notebook pravděpodobně nejvíce oblíbený mezi uživateli jazyka Python, vzniklo velké množství kernelů podporujících také další programovací jazyky (C, C++, C#, PHP, Matlab, Powershell, ruby, Brainfuck atd.).

Automaticky se při spuštění Jupyter Notebooku otevře i webový prohlížeč, ve kterém se samotný editor kódu nachází. Uživatel zde má možnost svůj kód organizovat na úrovně jednotlivých buněk. Zapsaný výraz (kód) je poté po stlačení kláves Shift+Enter vyslán na server, kde dojde k jeho zpracování, vyhodnocení a následnému zpětnému zaslání do webového prohlížeče. Prohlížeč následně pomocí JavaScriptu interpretuje zpětnou vazbu vyhodnoceného původního požadavku a správně ho umístí do dynamické webové stránky 4.

4.3 Knihovny

Jazyk Python je dnes vyhledáván především díky široké nabídce nástrojů na zpracování objemných datasetů, analýzu dat, strojové učení atd. Já ve své praktické části také několik typů balíčků využívám, proto je zde krátce představím a vysvětlím jejich funkci.

Matplotlib

Matplotlib je volně dostupná knihovna, která slouží jako nástroj pro vizualizaci a analýzu dat. Umožňuje komplexní vykreslování grafů, histogramů či teplotních map. Knihovna velmi dobře zvládá pracovat v prostředí Jupyter Notebook, proto také byla využita pro finální interpretaci vytvořených projekcí [11].

TKinter

Modul Tkinter je volně dostupná sada widgetů, jenž umožňuje v programovacím jazyce Python vytvářet základní uživatelký interface (GUI). Uživateli tak umožní vytvářet okna (obdobně jako v prostředí Windows), které mohou mít např. informativní účel nebo mohou řídit chod skriptu.

SimpleITK

SimpleITK slouží k práci s digitalizovaným obrazem a jeho analýzu. Opět se jedná o volně dostupný balíček, jenž je pouze zjednodušenou verzí knihovny ITK (Insight Segmentation and Registration Toolkit), ta je obvzlášť v oblasti registrace a zpracování lékařských obrazových dat velmi populárním nástrojem [17, 35].

OS

Abychom mohli v Pythonu interagovat se samotným systémem, využíváme modulu OS. Jedná se o jeden ze základních Python balíčků. OS poskytuje přístup k práci se souborovým systémem, procesy, plánovačem atd.

NumPy

„NumPy“ je v angličtině zkratkou pro výraz Numerical Python extension. Je to další ze základních balíčků jazyka Python. Uživatelům poskytuje matematické funkce, umožňuje rychle a efektivně pracovat s n-rozměrnými poli či maticemi. Je schopná pracovat s libovolnými datovými typy a snadno ji můžeme využít při práci s databázemi.

NiBabel

NiBabel je další z řady knihoven pro Python, jenž umožňuje práci s širokou škálou formátů obrazů. Především se ale zaměřuje na formáty standardně využívané ve zdravotnictví, kde zabezpečuje načítání dat, přístup k metadatům a zápis dat produkovaných např. magnetickou rezonancí. Podporovanými formáty jsou například NifTI-1, NifTI-2, DICOM atd. [1]

tqdm

Přítomnost modulu tqdm neovlivní samotný chod skriptu. V mém případě zastává spíše informativně-estetickou funkci. K importu tohoto balíčku jsem se rozhodla na základě poznatku, že zpracování objemových dat může mnohdy trvat i desítky minut, proto je vhodné upozornit uživatele pomocí stavového řádku na stav probíhajícího výpočtu a jeho úroveň vyhotovení [2].

Detecta

Balíček funkcí vytvořený Marcem Duartem k detekování různých případů v námi dodaných datech. Jedná se o velmi triviální a zároveň intuitivní funkce. V mém případě využívám funkci detect_peaks.py, jenž má stejné parametry a téměř konsistentní výsledky s funkcí findpeaks z balíčku Signal Processing Toolbox od Matlabu [6].

Pydicom

Pydicom byl vyvinut čistě pro potřeby Pythonu. Jak název napovídá, balíček pracuje s datovým formátem DICOM, uživateli tak umožňuje tato data načíst, modyfikovat a ukládat. I když balíček je schopný samostatného fungování, je velmi častou praxí, že je při práci s pixely kombinován s balíčkem NumPy [19].

Importování knihoven

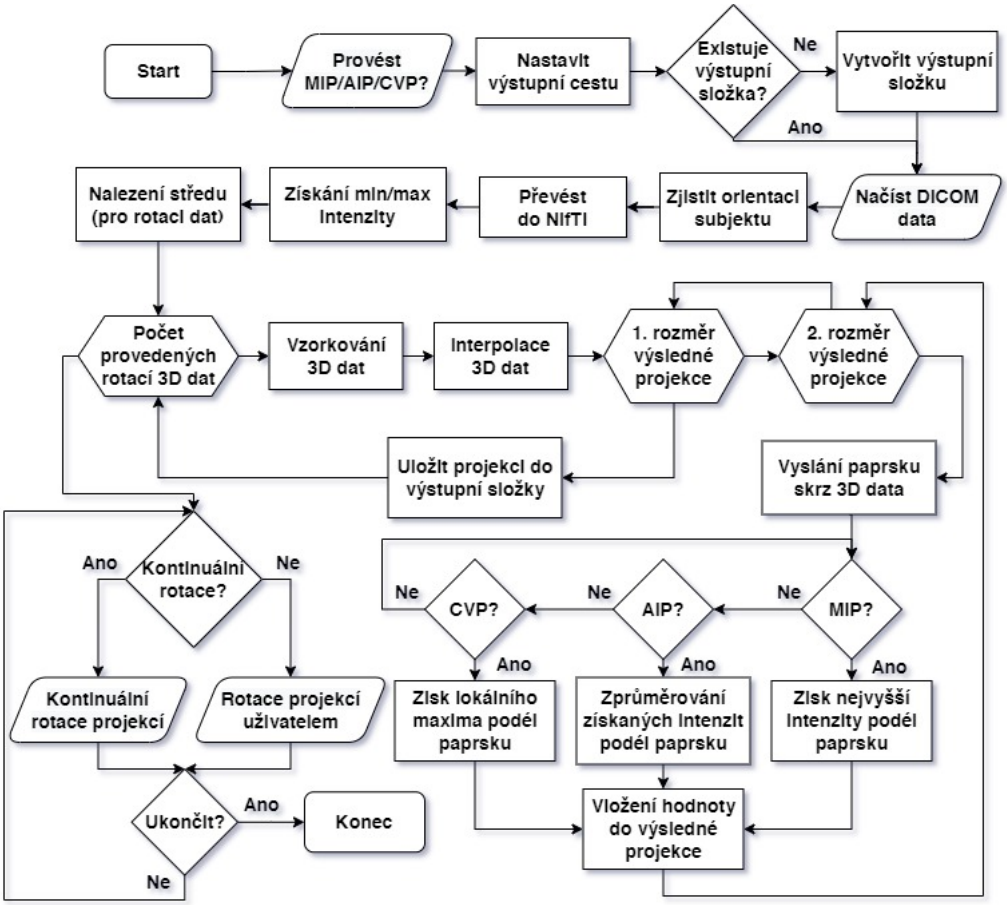
Výše zmíněné knihovny byly do skriptu importovány pomocí příkazu `import`. Z některých balíčků byly importovány pouze jednotlivé funkce, pro daný účel posloužil příkaz `from` (uvádím pouze pár vybraných příkladů pro ukázku):

```
1 import matplotlib
2 from detecta import detect_peaks
3 import pydicom as dicom
```

Výpis 4.1: Import knihoven v Pythonu

5 Návrh algoritmu a zdroj dat

5.1 Vývojový diagram



Obr. 5.1: Navržený algoritmus pro zobrazení 3D dat

5.2 Zdroj dat

Pro návrh algoritmu a jeho následnou implementaci jsem potřebovala volně dostupnou medicínskou databázi. Data z této databanky bylo třeba řádně prostudovat a následně zpracovat. Po prozkoumání několika volně dostupných medicínských databází jsem zvolila databanku projektu The Visible Human Project na stránkách Magnetic Resonance Research Facility spravovanou University of Iowa[1].

Zmíněná databanka nabízí CT snímky různých částí těla muže i ženy. Nasnímané data jsou uložena v souborech typu DICOM [38].

[1]Dostupné z: https://mri.radiology.uiowa.edu/visible_human_datasets.html

5.3 Vstupní datasety DICOM

Ze zmíněné databanky v 5.2 byly nakonec využity celkem čtyři datasety, přesněji datasety Ankle, Head, Pelvis a Shoulder (česky Kotník, Hlava, Pánev a Rameno).

Datasety obsahují následující hodnoty základních parametrů (více o jednotlivých parametrech uvádí [39]):

Tab. 5.1: Společné parametry DICOM datasetů

Atribut	Štítek	DICOM data
Image Type	(0008,0008)	Original, Primary, Axial
Modality	(0008,0060)	CT
Series Description	(0008,103E)	Resampled to 1mm voxels
Patient's Sex	(0010,0040)	F
Slice Thickness	(0018,0050)	1
Patient Position	(0018,5100)	HFS
Image Orientation (Patient)	(0020,0037)	[1,0,0,0,1,0]
Photometric Interpretation	(0028,0004)	MONOCHROME2
Pixel Spacing	(0028,0030)	[1, 1]
Bits Allocated	(0028,0100)	16
Bits Stored	(0028,0101)	16

Tab. 5.2: Rozlišení DICOM datasetů

Atribut	Štítek	Ankle	Head	Pelvis	Shoulder
Rows	(0028,0010)	512	512	512	512
Columns	(0028,0011)	512	512	512	512
Počet řezů	-	150	234	150	450

6 Implementace Algoritmu

6.1 Úvodní nastavení

Projekce

Po prvotním spuštění skriptu je uživatel widgetem s názvem *Settings* vyzván, aby definoval požadovaný typ projekce, jenž bude následně vytvořen. Nabídka se skládá ze tří typů projekcí (obrázek A.1):

- Maximum Intensity Projection
- Average Intensity Projection
- Closest Vessel Projection

Uživateli je umožněno si vybrat pomocí checkbuttons zároveň více typů projekcí. Výběr je třeba následně potvrdit tlačítkem *Apply settings*. Také je ošetřena událost, kdy uživatel nezvolí ani jednu možnost, v takovém případě není umožněn postup dále a widget *Settings* se zobrazí uživateli znovu.

Výstupní cesta

Dalším krokem v algoritmu je nastavení výstupní cesty. Uživatel je nejdříve upozorněn informativním widgetem (obrázek A.2) a následně po odkliknutí tlačítka *OK* se zobrazí dialogový box pro volbu požadované cesty.

Na místě, na které se odkazuje tato výstupní cesta, dojde ke kontrole, zdali již na této adrese existuje složka *OUTPUT_PROJECTION*. Pokud se zde taková složka nenachází, dojde k jejímu vytvoření (uživatel je v konzoli informován vytvoření výstupní složky). Existuje-li výše zmíněný adresář, je předchozí krok vynechán.

Adresář *OUTPUT_PROJECTION* je obvzlášť důležitý v následujících krocích algoritmu, jelikož se zde budou ukládat vytvořené 2D projekce.

Orientace DICOM dat

Informace o orientaci je stěžejní ke správnému zobrazení vytvářené 2D projekce.

Podobně jako v podkapitole 6.1 je uživatel nejdříve upozorněn informativním widgetem, aby vybral adresář obsahující celý dataset DICOM souborů (obrázek A.3).

Funkce `get_orientation()` dle dostupných metadat zjistí informace o způsobu snímání dodaných dat. Jak naznačuje výpisek z funkce 6.1, nejdříve dojde k načtení metadat z prvního souboru DICOM datasetu pomocí příkazu `dicom.dcmread()`, dále funkce ověří, jestli je v metadatech zapsán atribut *PatientPosition*. Pokud ano, je tato informace přiřazena do proměnné `orientation`. Pokud záznam neexistuje

(data byla například poškozena), je uživateli zobrazena nabídka s výběrem podporovaných orientací subjektu při snímání (obrázek A.4).

Algoritmus je schopný zobrazit všechny orientace uvedené ve zmíněné nabídce `shortcut`. Jedná se o běžné typy natočení lidského pacienta, který je snímán v pozicích na břiše (Prone), na zádech (Supine), na pravém či levém boku (Decubitus), ve směru od hlavy k chodidlům (Head First) anebo naopak (Feet First).

Pokud by atribut nabýval jiných hodnot, případně by uživatel zvolil nabídku *Other*), dojde k automatickému ukončení skriptu. Samozřejmě i jiné typy natočení se u snímaných objektů vyskytují, nicméně se zpravidla jedná o drobné živočichy (např. myši). Já jsem ve své bakalářské práci primárně pracovala s datasety obsahující lidské subjekty, proto jsem algoritmus o tyto orientace dále nerozšiřovala.

```
1 if hasattr(ds, 'PatientPosition'):
2     orientation = ds.PatientPosition
3     shortcut = ["HFP","HFS","HFDR","HFDL","FFDR",
4                 "FFDL","FFP","FFS"]
5
6     if orientation not in shortcut:
7         oritentation = "other"
```

Výpis 6.1: Ukázka z funkce get_orientation()

Převod do 3D podoby

Proces konverze formátů využívá funkci `ImageSeriesReader` z knihovny SimpleITK. Reader si nejdříve načte funkcí `GetGDCMSeriesIDs` ID zpracovávané DICOM série. Identifikátor dále využije jako unikání hodnotu souborů dané série. Dle ID si ověří, že soubor skutečně patří do série a poté si uloží jeho název, to vše provede pomocí funkce `GetGDCMSeriesFileNames`. Zmíněná metoda také zaručí správné pořadí načtení jednotlivých řezů (souborů DICOM).

Druhý reader poté využije uložené názvy souborů, načte metadata souborů a pomocí příkazu `Execute()`, převede DICOM dataset do jednoho DICOM souboru, kde jsou data uložena ve 3D formě (doteď jsme se pohybovali na 2D úrovni jednotlivých řezů).

6.2 Vytváření projekce

Samotný proces vytváření projekcí obstarává v algoritmu funkce `projection()`. Na vstupu funkce je vytvořený 3D soubor, název DICOM datasetu, výstupní cesta,

informace o požadovaném typu projekce (případně požadovaných typech projekcí) a již výše zmiňovaná stěžejní informace o orientaci subjektu při snímání.

Minimální a maximální intenzita voxelů

Hodnoty minimální a maximální intenzity obrazu lze získat využitím obrazového filteru knihovny SimpleITK. I když označení obrazový filter se zdá z počátku zavádějící, funkce `MinimumMaximumImageFilter` po nastavení příkazem `Execute()` dokáže získat obě hraniční hodnoty, ty se poté jednoduše uloží do proměnných přes funkce `GetMinimum()` a `GetMaximum()`.

Střed obrazu a rotace obrazu

Jelikož k práci s volumetrickými daty jsem zvolila knihovnu SimpleITK, bylo třeba si uvědomit způsob zacházení s daty touto knihovnou. SimpleITK definuje obraz jako sadu bodů na mřížce (anglicky „grid“), které zároveň zaujímají určitý fyzický prostor. Tento základní princip knihoven ITK se výrazně liší od mnoha jiných knihoven určených k práci s obrazy, ty běžně zacházejí s obrazy jako s polem pixelů či voxelů s izotropní vzdáleností a zároveň nepracují s fyzickým prostorem, do kterého by měl být obraz zasazen. Oblast fyzického prostoru je v ITK definována parametry:

- **Origin** (původ) - lokace voxelu v souřadnicovém systému
- **Spacing** (vzdálenost) - vzdálenost mezi jednolivými pixely v rámci jedné dimenze
- **Size** (velikost) - počet pixelů v rámci jedné dimenze
- **Direction cosine matrix** (matice směrových kosinů) - členy matice vyjadřují kosinus úhlu mezi osou pootočené soustavy souřadnic a osou původní soustavy souřadnic

Proto aby došlo k rotaci dat a následnému vytvoření projekcí, bylo třeba nejdříve nalézt souřadnice středu volumetrických dat, následně zjistit jeho hodnoty v oblasti fyzického prostoru a teprve poté bylo možné okolo nalezeného středu data otáčet. Proces nalezení fyzického středu zajišťuje funkce `find_image_center()`.

Samotná rotace poté probíhá pro úhly v rozsahu 0° až 359°. K rotaci dat využívám další funkci z balíčku SimpleITK a to `Euler3DTransform()`. Funkce aplikuje na data 3D rigidní transfomaci rotací okolo fixního bodu (středu) a zároveň nabízí i translaci dat. Jejími parametry jsou tři eulerovské úhly (zadáváme v radiánech) definující rotaci okolo příslušných os (x, y a z) a další tři parametry definují translaci v příslušných dimenzích.

Jak již bylo zmíněno výše, rotace pomocí eulerovských úhlů probíhá okolo základních os x, y a z. Tyto úhly jsou označeny písmeny ψ, θ a ϕ.

Rotace o ψ radiánů okolo x osy je definovaná následovně:

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix} \tag{6.1}$$

Obdobně můžeme zadefinovat i rotaci o θ radiánů okolo y osy:

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{6.2}$$

A na závěr doplnění rotace o $+\phi$ radiánů kolem z osy:

$$R_z(\phi) = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.3}$$

Výsledná rotační matice je sekvencí tří výše zmíněných rotací (jedna rotace připadá na jednu základní osu). Jelikož násobení matic není komutativní, pořadí rotací je pro finální výsledek zásadní. Nicméně v mém případě otáčím data pouze okolo osy z, tudíž do nastavených parametrů funkce `Euler3DTransform()`, přijde na pozice ψ a θ 0 a na místo ϕ bude dosazena aktuální hodnota požadovaného natočení dat (převedena ze stupňů na radiány).

Dále je potřeba funkcí `Euler3DTransform()` zadefinovat souřadnice fyzického středu. Poté pomocí příkazu `Resample()` dojde k převzorkování (natočení) dat. Vstupními parametry `Resample()` jsou obraz, jenž bude rotován, referenční (původní) obraz, nastavená požadovaná rigidní transformace, požadovaný typ interpolace dat a defaultní hodnota obrazu.

2D projekce

Proces vytvoření 2D projekce volumetrických dat je téměř identický pro Maximum Intensity Projection a Average Intensity Projection. Postup u Closest Vessel Projection se liší ve způsobu výpočtu výsledných hodnot projekce, nicméně průběh vytvořené funkce je opět velmi podobný dvěma výše zmíněným metodám.

U všech uvedených metod je nejdříve zadefinováno 2D pole o příslušných rozměrech volumetrických dat ve směru os x a z. Vytvořené pole je vyplněno hodnotou, jenž se rovná minimální intenzitě volumetrických dat (ta značí prázdný prostor).

Algoritmus následuje dvojicí **`for`** cyklů, které mají za úkol projít 3D obraz v příslušném natočení a získat výstupní hodnoty podél osy y, které se budou ukládat do vyindexovaného místa výstupní 2D projekce.

Jak již bylo zmíněno v kapitole 2.1.3, metoda Maximum Intensity Projection hledá podél osy y maximální hodnotu (funkce `np.amax()`), kterou následně vyhodnotí jako výstupní pro příslušné indexy zbylých dvou os. Pro Average Intensity Projection použijeme identický postup, avšak funkce k výpočtu výsledné hodnoty se bude lišit (funkce `np.mean()`).

Pro metodu Closest Vessel Projection je postup doplněn o nastavení prahové hodnoty (obrázek A.5), dle které bude následně funkce vyhodnocovat lokální maxima podél osy y. K nalezení lokálních maxim přesahujících požadovanou prahovou hodnotu slouží funkce `detect_peaks()`. Na výstupu této funkce je vektor pozic všech lokálních maxim podél osy y, jenž splnila zadané podmínky. Do výstupní 2D projekce přijde příslušná hodnota odkazující se pomocí indexů zbylých os a první hodnoty vektoru pozic.

Uložení vytvořené projekce

Algoritmus každou vytvořenou 2D projekci pro definované natočení volumetrických dat uloží. Zde se také uplatní výše zmíněná orientace subjektu při snímání. V případě, že byla data snímána v režimu Head First, byly by všechny dosavadně vytvořené projekce vzhůru nohama. Proto je ve funkci `save_projection()` **`if`** podmínka kontrolující orientaci subjektu. Pokud byla dodaná data skutečně snímána Head First, dojde k otočení projekce o 180°.

Dále je vytvořeno příslušné jméno projekce a samotná projekce je pomocí funkce `WriteImage()` zapsaná jako komprimovaný soubor NifTI (.nii.gz) do složky *OUTPUT_PROJECTION*.

6.3 Zobrazení projekcí

Poslední částí implementace návrženého algoritmu je zobrazení vytvořených projekcí. Celou část spravuje funkce `set_projections_and_plotting()`. Ta uživatele v případě více vytvořených projekcí vyzve k volbě preferované projekce k vykreslení. Uživatel je dále pomocí informativního widgetu (obrázek A.6) upozorněn na typ aktuálně připravované projekce, která se bude zobrazovat.

Následuje možnost výběru mezi požadovaným typem zobrazení dané projekce (obrázek A.7). Zde má uživatel na výběr mezi dvěma nabídkami:

- Continous rotation
- Scroll for a rotation

Funkce `set_projections_and_plotting()` funguje jako nekonečná smyčka, tzn. dokud si uživatel bude přát zobrazovat vytvořené projekce, bude skript neustále nabízet možnost jejich zobrazení (obrázek A.8). Pokud se uživatel rozhodne skript ukončit

a zvolí *No*, program uživateli oznámí, že se nachází na konci skriptu (obrázek A.9) a dojde k ukončení.

Magic funkce

IPython a také nástroj Jupyter Notebook nabízí kolekci několika předdefinovaných funkcí, které označuje jako *magic functions*. Tyto funkce mohou být zavolány stejnou syntaxí jako běžné příkazy. Aby algoritmus zajistil interaktivitu vykreslených projekcí, bylo třeba do skriptu zakomponovat tzv. *line magic function*. Ve skriptu se přesněji nachází matplotlib magic funkce, ta ovlivní způsob vykreslování grafů a obrazů vizualizovaných knihovnou Matplolib 6.2. GUI je označení backendu knihovny Matplotlib, jenž bude spuštěn po zavolání funkce, příklady backendů: inline, notebook, qt atd.

```
1 %matplotlib [gui]
```

Výpis 6.2: Line magic function

Kontinuální načítání mezi projekcemi

Při volbě *Continuous rotation* (obrázek A.7) jsou jednotlivé projekce kontinuálně načítány a se zpožděním 0,5 sekundy vykreslovány do grafu. Dochází k postupnému překreslování aktuálně zobrazené projekce projekcí následující, tím je docílena pomyslná, neustálá rotace obrazu.

Funkce běží na principu **while** smyčky, dokud si přeje uživatel vykreslovat vytvořené projekce, skript načítá projekce 0° až 359° neustále dokola. Funkce je ukončena až akcí uživatele, k tomu je vyžadováno kliknutí do vnitřního prostoru vykreslovaného grafu levým tlačítkem myši (obrázek A.10). Tato akce je zaznamenána a vyhodnocena jako žádost o ukončení vykreslování 6.3.

```
1 if ('button' in mutable_object.keys()
2     and True in mutable_object.values()):
3     plt.close(fig=None)
4     break
```

Výpis 6.3: Podmínka k ukončení vykreslování projekcí

Uživatelem definované natočení projekce

Pokud volba uživatele bude příslušet možnosti *Scroll for a rotation* (obrázek A.7), skript spustí postupně dvě funkce obstarávající tento typ vykreslení.

Nejdříve je zavolána funkce `stack_projections()`. Ta zajistí postupné načtení všech vytvořených 2D projekcí a jejich zařazení do fronty (anglicky „queue“), která funguje na principu FIFO (First-In-First-Out), tedy první prvek vstupující do fronty z ní také vychází první. Aby skript mohl vykreslovat projekce neustále dokola, byla fronta modifikována, tak aby z ní po prvotní inicializi žádný prvek nevystupoval a zároveň do ní ani nevstupoval, tím došlo k vytvoření cyklické fronty (anglicky „circular queue“).

Vytvořenou frontu tak ve finále tvoří 360 projekcí poskládaných za sebe. První načtená projekce inicializuje nejdříve 2D frontu `stacked_array`, ta je dále rozšířena o třetí rozměr funkcí `np.newaxis`. Každá další načtená projekce je rozšířena třetí dimenzí a následně je zařazena do fronty za naposledy zařazenou projekci.

Po vytvoření fronty je spuštěna druhá funkce `slice_viewer()`. Ta je zároveň kontrolní funkcí pro interaktivní vykreslení projekcí, které je schopno reagovat na požadavek uživatele, aby došlo k „otočení“ o určitý úhel. Z předchozích odstavců je zřejmé, že k reálnému otočení nedochází, nicméně je zajištěna změna hodnoty indexu, jenž odkazuje na pozici ve frontě projekcí. Index tak určí, která projekce se bude uživateli vykreslovat. Graf je opět schopný zaznamenat události spuštěné myší, avšak tentokrát vyhodnocuje posouvání prostředního kolečka (obrázek A.11).

Graf je pomocí funkce `fig.canvas.mpl_connect()` schopný zaznamenat zmíněné události a vyhodnotit, zdali došlo k posunu kolečka nahoru nebo dolů 6.4. Od toho se odvíjí další nastavení zobrazení.

```
1  fig = event.canvas.figure
2  ax = fig.axes[0]
3
4  if event.button == 'up':
5      previous_slice(ax)
6
7  elif event.button == 'down':
8      next_slice(ax)
9
10 fig.canvas.draw()
```

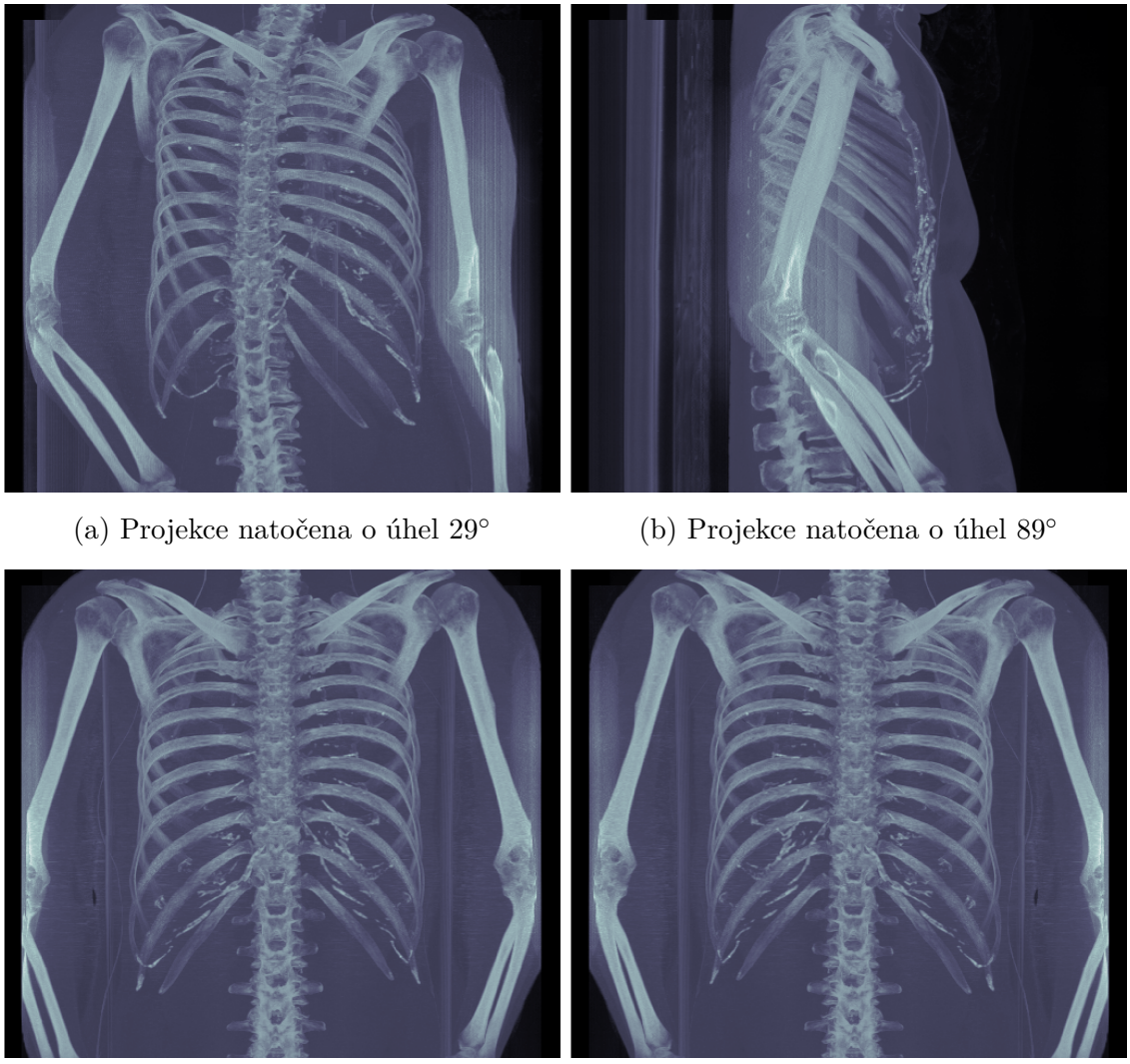
Výpis 6.4: Funkce zaznamenávající a vyhodnocující události

V případě posunu kolečka směrem nahoru, dojde k dekrementaci indexu fronty projekcí o 1 a vykreslovaný graf se překreslí projekcí s příslušným indexem. Pokud dojde k posunu kolečka směrem nahoru, index se o 1 inkremetuje a opět dojde k překreslení aktuální projekce.

7 Výstupní data

Vygenerované obrázky výstupních projekcí pochází ze všech zakomponovaných metod v algoritmu (MIP, AIP, CVP). Aby byl výstup každé metody názorněji zobrazen a dal se lépe rozlišit od metod jiných, byly obrázky v rámci jednoho datasetu pokaždé generovány se stejným natočením.

7.1 Vygenerované projekce



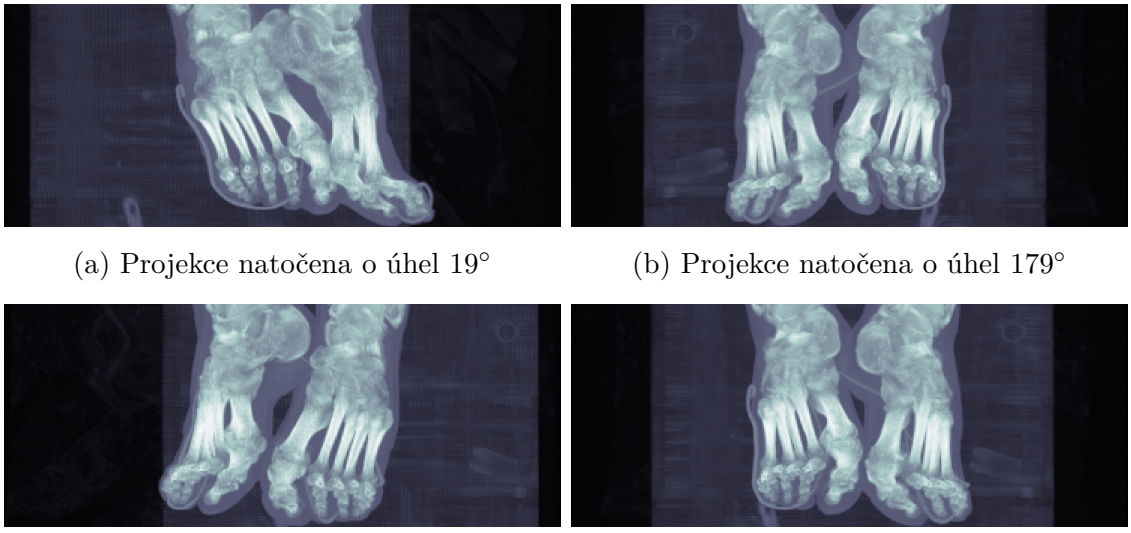
(a) Projekce natočena o úhel 29°

(b) Projekce natočena o úhel 89°

(c) Projekce natočena o úhel 179°

(d) Projekce natočena o úhel 359°

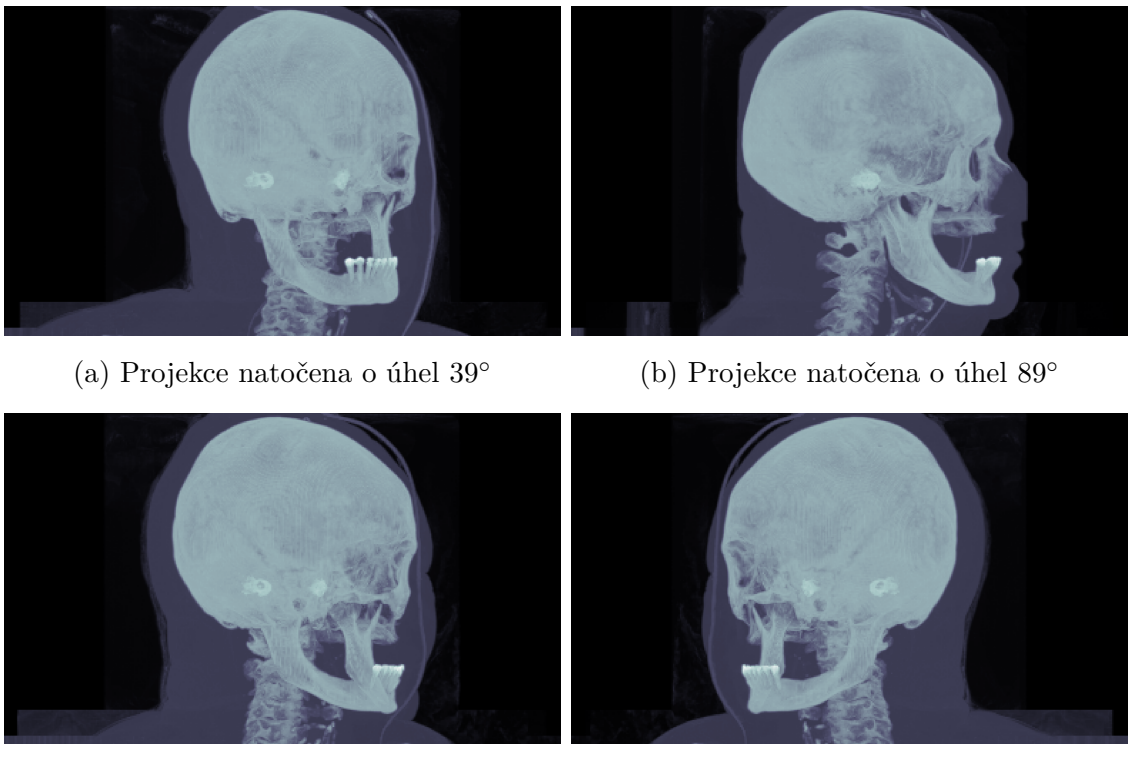
Obr. 7.1: Metoda MIP aplikovaná na DICOM dataset Shoulder



(a) Projekce natočena o úhel 19° (b) Projekce natočena o úhel 179°

(c) Projekce natočena o úhel 339° (d) Projekce natočena o úhel 359°

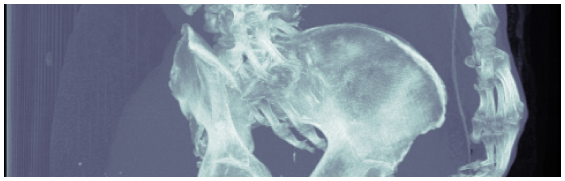
Obr. 7.2: Metoda MIP aplikovaná na DICOM dataset Ankle



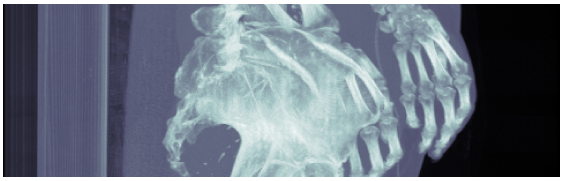
(a) Projekce natočena o úhel 39° (b) Projekce natočena o úhel 89°

(c) Projekce natočena o úhel 129° (d) Projekce natočena o úhel 319°

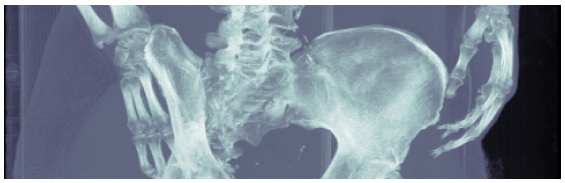
Obr. 7.3: Metoda MIP aplikovaná na DICOM dataset Head



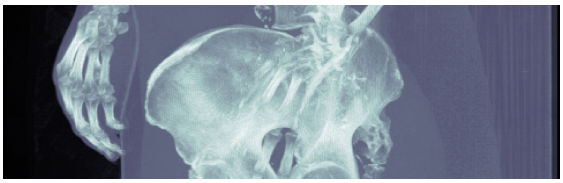
(a) Projekce natočena o úhel 49°



(b) Projekce natočena o úhel 99°



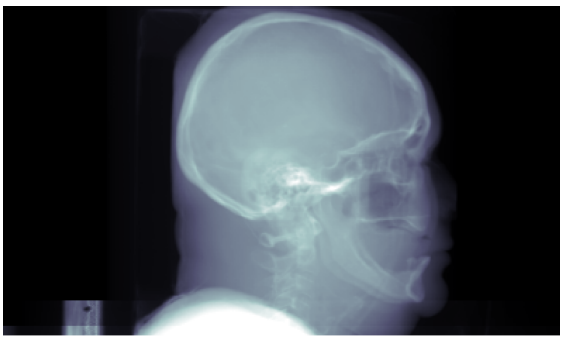
(c) Projekce natočena o úhel 149°



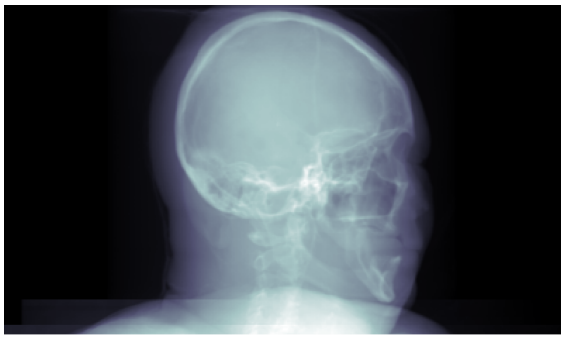
(d) Projekce natočena o úhel 299°

Obr. 7.4: Metoda MIP aplikovaná na DICOM dataset Pelvis

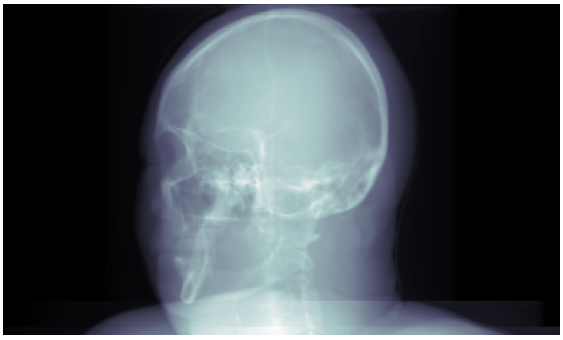
(a) Projekce natočena o úhel 39°



(b) Projekce natočena o úhel 89°

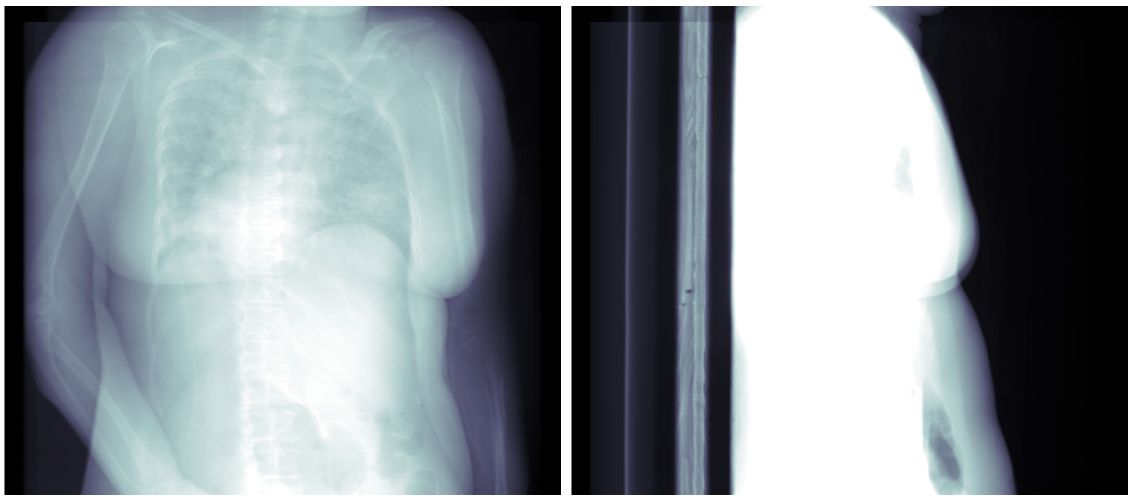


(c) Projekce natočena o úhel 129°



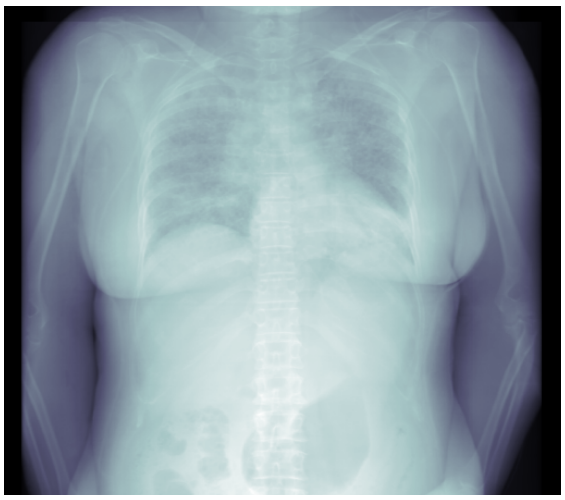
(d) Projekce natočena o úhel 319°

Obr. 7.5: Metoda AIP aplikovaná na DICOM dataset Head

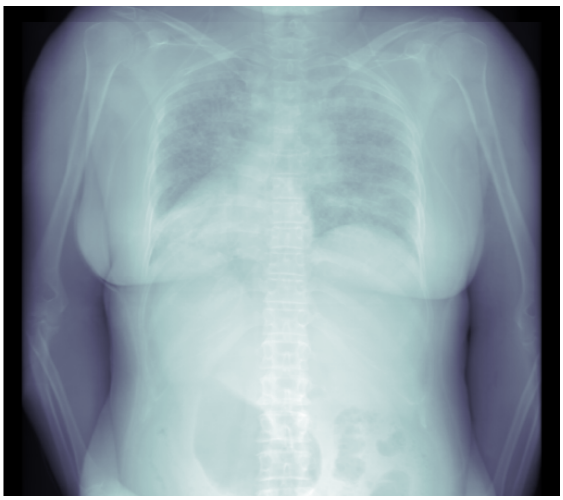


(a) Projekce natočena o úhel 29°

(b) Projekce natočena o úhel 89°

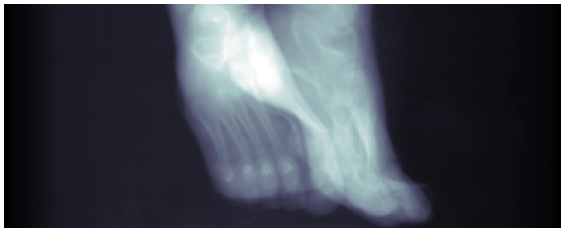


(c) Projekce natočena o úhel 179°

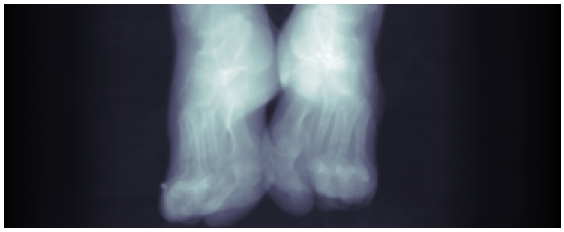


(d) Projekce natočena o úhel 359°

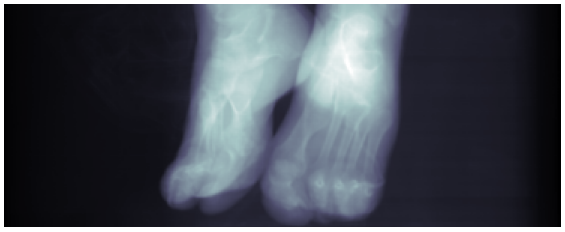
Obr. 7.6: Metoda AIP aplikovaná na DICOM dataset Shoulder



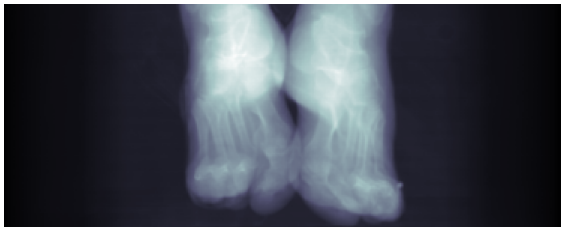
(a) Projekce natočena o úhel 19°



(b) Projekce natočena o úhel 179°



(c) Projekce natočena o úhel 339°

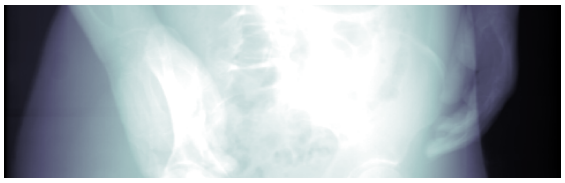


(d) Projekce natočena o úhel 359°

Obr. 7.7: Metoda AIP aplikovaná na DICOM dataset Ankle

(a) Projekce natočena o úhel 49°

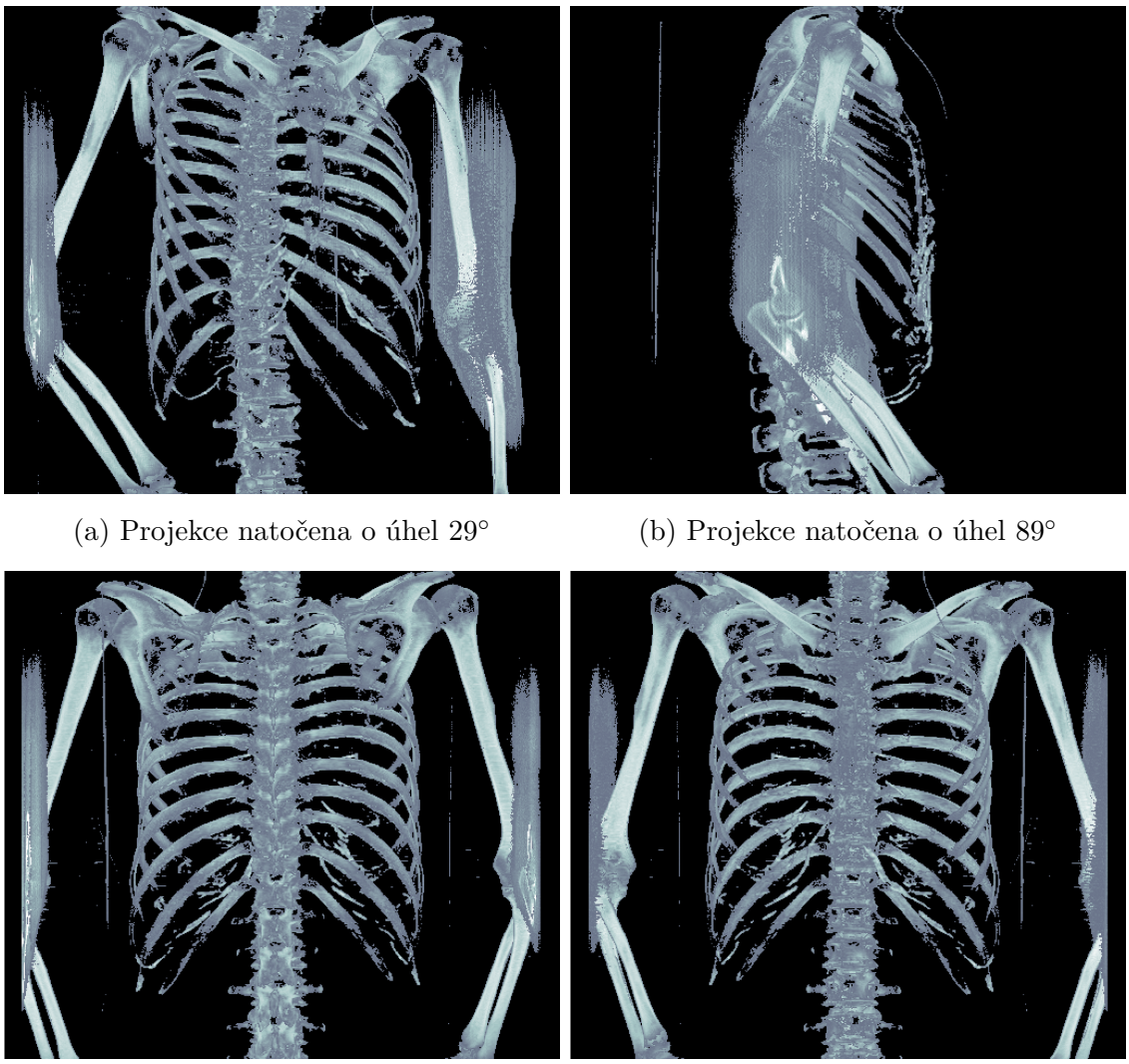
(b) Projekce natočena o úhel 99°



(c) Projekce natočena o úhel 149°

(d) Projekce natočena o úhel 299°

Obr. 7.8: Metoda AIP aplikovaná na DICOM dataset Pelvis



(a) Projekce natočena o úhel 29°

(b) Projekce natočena o úhel 89°

(c) Projekce natočena o úhel 179°

(d) Projekce natočena o úhel 359°

Obr. 7.9: Metoda CVP aplikovaná na DICOM dataset Shoulder

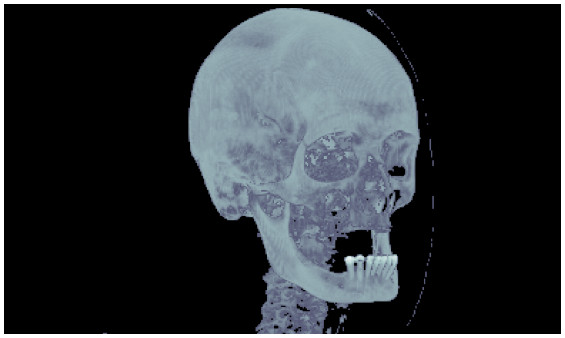
(a) Projekce natočena o úhel 19°

(b) Projekce natočena o úhel 179°

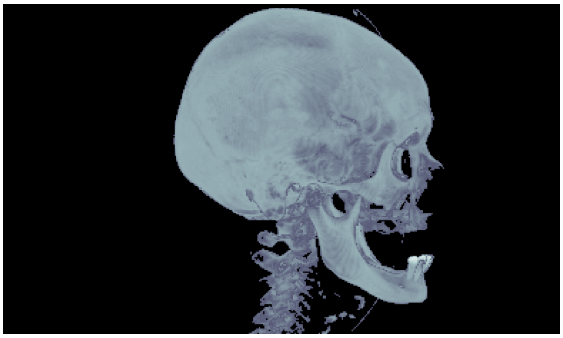
(c) Projekce natočena o úhel 339°

(d) Projekce natočena o úhel 359°

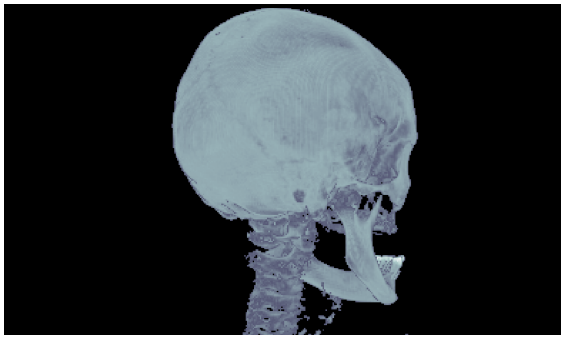
Obr. 7.10: Metoda CVP aplikovaná na DICOM dataset Ankle



(a) Projekce natočena o úhel 39°



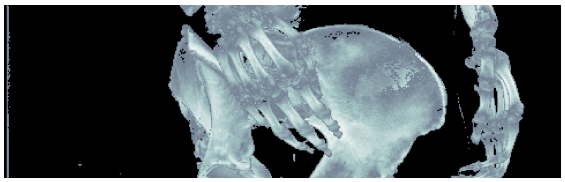
(b) Projekce natočena o úhel 89°



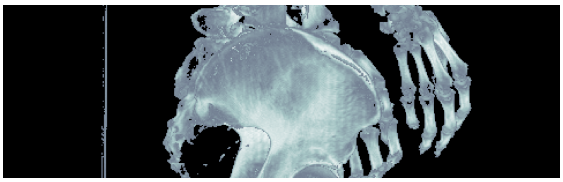
(c) Projekce natočena o úhel 129°

(d) Projekce natočena o úhel 319°

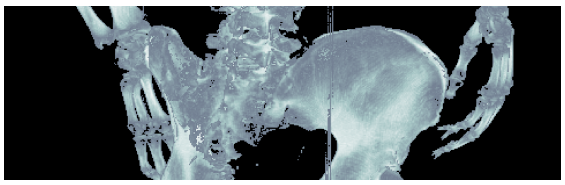
Obr. 7.11: Metoda CVP aplikovaná na DICOM dataset Head



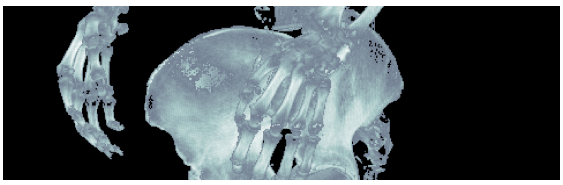
(a) Projekce natočena o úhel 49°



(b) Projekce natočena o úhel 99°

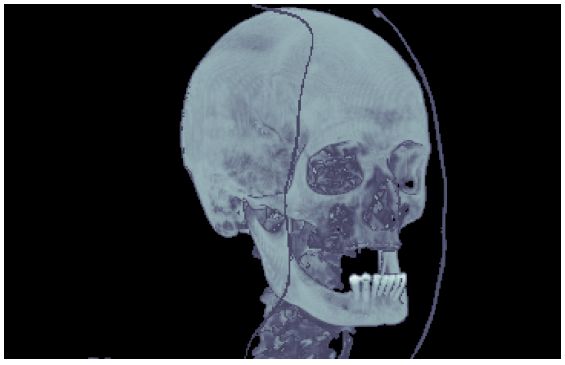


(c) Projekce natočena o úhel 149°

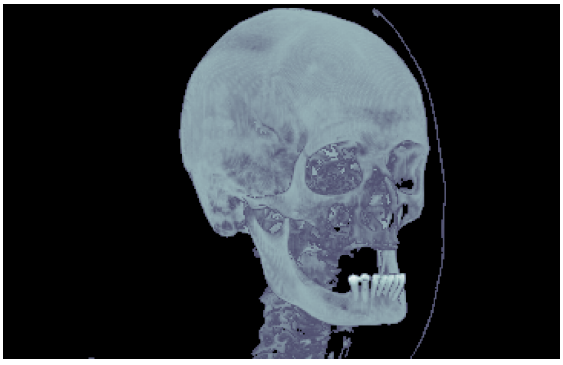


(d) Projekce natočena o úhel 299°

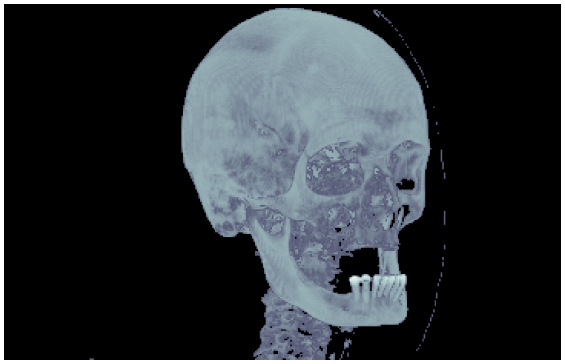
Obr. 7.12: Metoda CVP aplikovaná na DICOM dataset Pelvis



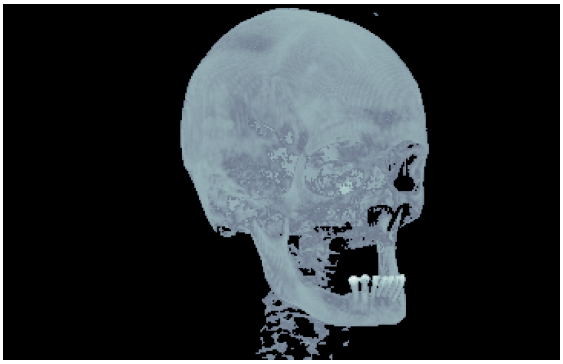
(a) Prahová hodnota intenzity 200



(b) Prahová hodnota intenzity 450



(c) Prahová hodnota intenzity 580



(d) Prahová hodnota intenzity 1000

Obr. 7.13: Metoda CVP aplikovaná na DICOM dataset Head za použití různých prahových hodnot

8 Hodnocení algoritmu a výstupních dat

Poslední kapitolou mé praktické části je testování funkčnosti vytvořeného programu a zároveň okomentování jeho výpočetní a paměťové náročnosti.

8.1 Konfigurace testovacího zařízení

Průběh algoritmu je samozřejmě ovlivněn i využitými komponenty v zařízení, na kterém je spouštěn. Proto je vhodné uvést konfiguraci testovacího zařízení.

K vyhodnocení skriptu bylo využito notebooku s operačním systém Windows 10 Pro (64bitový), nainstalovanou operační pamětí o velikosti 16 GB, s procesorem Intel® Core™ i7-9750H CPU @ 2.60 GHz, integrovanou grafickou kartou Intel® UHD Graphics 630 a dedikovanou grafickou kartou NVIDIA Quadro T1000 (4 GB).

8.2 Výsledky testování algoritmu

Testování probíhalo pro všechny využité datasety (popsáno v kapitole 5.3) a to pro každou metodu zvlášť (Maximum Intensity Projection, Average Intensity Projection a Closest Vessel Projection). Při testování byly sledovány tři parametry:

- průměrné zatížení centrální procesorové jednotky (CPU)
- průměrné zatížení operační paměti (RAM)
- časová náročnost

K testování bylo využito knihoven tqdm a Psutil (nabízí rozhraní pro zisk údajů o procesech a využití systému, např. CPU, paměť, disky, síť atd.). Výsledky byly současně porovnávány s informacemi, jež jsou podávány systémovým Správcem úloh a rozšiřujícím programem Sledování prostředků.

Každé testování proběhlo opakovaně za různých podmínek (skript byl spuštěn poprvé či byl již předtím spuštěn několikrát) a finální hodnota udávaná v tabulkách 8.1, 8.2 a 8.3 je průměrem naměřených hodnot.

Tab. 8.1: Výsledky testování Maximum Intensity Projection

Maximum Intensity Projection			
Dataset	CPU [%]	RAM [MB]	Čas [h:mm:ss]
Ankle	22,59	641	0:05:28
Head	23,07	936	0:08:32
Pelvis	22,69	778	0:05:33
Shoulder	22,64	1751	0:16:35

Tab. 8.2: Výsledky testování Average Intensity Projection

Average Intensity Projection			
Dataset	CPU [%]	RAM [MB]	Čas [h:mm:ss]
Ankle	18,31	633	0:07:57
Head	18,86	874	0:12:24
Pelvis	18,66	644	0:08:52
Shoulder	19,16	1599	0:24:25

Tab. 8.3: Výsledky testování Closest Vessel Projection

Closest Vessel Projection			
Dataset	CPU [%]	RAM [MB]	Čas [h:mm:ss]
Ankle	11,51	616	0:26:18
Head	11,72	874	0:39:14
Pelvis	11,67	637	0:26:24
Shoulder	11,62	1524	1:15:35

Operační paměť (RAM)

V případě hodnocení zatížení RAM byla opět nejnáročnější metoda MIP. Avšak výsledky jednotlivých metod se v tomto bodě testování mezi sebou natolik nelišily. Spíše je vhodné poukázat na poměrně výrazný nárůst zatížení paměti u datasetu Shoulder. Vysvětlení tohoto rozdílu nabízí podkapitola 5.3, přesněji tabulka 5.2, jedná se totiž o nejrozsáhlejší dataset. Proto mohu v tomto bodě konstatovat, že výpočetní náročnost prudce poroste při nárůstu rozlišení vstupního datasetu.

Časová náročnost

Posledním bodem samotného testování bylo vyhodnocení časové náročnosti použitých metod. V této kategorii byly celkem jednoznačně nejkratší časy potřebné k vyhotovení projekcí u metody MIP, což tuto metodu vyzdvihuje v porovnání s výsledky z předchozích bodů.

Naopak doba běhu algoritmu výrazně narostla u CVP. Např. u zmíněného datasetu Shoulder se průměrná doba pohybovala okolo 1 hodiny a 19 minut.

Do testování CVP bylo také zakomponováno nastavení různých prahů, nicméně tento bod nijak výrazně neovlivnil chod algoritmu a výsledky jeho testování.

8.3 Hodnocení výstupních dat

Maximum Intensity Projection

Metoda MIP poměrně spolehlivě zachytila kosterní strukturu vstupních dat.

Problém nastává v určitých úhlech projekcí, kde je velmi náročné rozlišit popředí od pozadí zobrazované struktury. Dochází tak ke ztrátě prostorové informace.

Na vygenerovaných obrázcích jsou také vidět okolní struktury, aby došlo k jejich odstranění a zobrazení přesnějšího výstupu, bylo by vhodné před samotnou metodou MIP aplikovat segmentaci dat, kde by došlo k odstranění nepotřebného okolí.

Average Intensity Projection

Kvalita výstupu metody AIP je velmi odlišná pro jednotlivé datasety a zároveň různé úhly natočení. V případě natočení dat, kdy se mnoho vnitřních struktur nenachází za sebou, jsou snímky poměrně přesné. Opět se zde nabízí využití segmentace, která by výstup více zpřesnila.

Oproti MIP si můžeme povšimnout, že vygenerované obrázky jsou vyhlazenější a neobsahují mnoho ostrých hran.

Velikým záporem metody AIP je problematické zobrazení v momentech, kdy je více struktur za sebou. V takovém případě dojde k nahromadění informací o intenzitě, výstup je tak přeexponován a je zcela nepoužitelný z pohledu informace, kterou nám nabízí. Navíc i zde podobně jako u MIP dochází ke ztrátě prostorové informace.

Closest Vessel Projection

Pro zobrazení i méně výrazných struktur, jenž se nachází v popředí rotovaných dat, se nejlépe uplatnila metoda CVP. Přestože stále hledá výrazné hodnoty podél vyslaného paprsku, oproti MIP je schopná zachytit některé struktury daleko detailněji.

I zde by zcela jistě pomohl k lepším výsledkům pre-processing ve formě segmentace dat, nicméně výstup metody samotné se dá velmi ovlivnit nastaveným prahem. Jak je vidět na vygenerovaných obrázcích, pomocí prahování se metodě podařilo zbavit většiny nechtěných, okolních struktur (avšak některé místa i nadále vykazují jistou formu zašumění) a zobrazit pouze oblasti splňující podmínku dostatečné intenzity.

Nevýhodou metody je zcela jistě nutná znalost prahu intenzity, který je potřeba nastavit pro daný typ dat.

Závěr

Mým cílem bylo v rámci bakalářské práce provést rešerše základních přístupů používaných pro zobrazení objemových dat se zaměřením na přímé zobrazení, tzv. Direct Volume Rendering. Z metod přímého zobrazení jsem prostudovala přístupy jak triviální tak pokročilé. Podrobněji jsem si nastudovala právě principy metod triviálních, a to přesněji metody Maximum Intensity Projection, Average Intensity Projection a Closest Vessel Projection.

Dále jsem se zaměřila na procesy zpracování vícerozměrných obrazů, jenž slouží k přípravě dat k vizualizaci. Z pre/post-processingu jsem se věnovala několika jednoduše implementovatelným metodám segmentace a také popisu možností transformace dat.

V praktické části jsem postupně řešila návrh zmiňovaného algoritmu pro zobrazení dat, implementaci v jazyce Python a také jeho testování zároveň s hodnocením finálního výstupu algoritmu.

Samotným výstupem implementovaného algoritmu jsou vygenerované 2D projekce. Několik příkladů těchto projekcí ve své práci také prezentuji. Na zmíněných projekcí si např. můžeme všimnout, že metody MIP a AIP ztrácí prostorovou informaci. Dále je u projekcí typu AIP potřeba respektovat úhel natočení dat. V případech, kdy se ve směru paprsku nacházelo za sebou více výrazných strukur, došlo k znehodnocení výsledné projekce nahromaděním informace o intenzitě podél vyslaného paprsku. Dalším významným poznatkem je závislot metody CVP na nastavené prahové hodnotě. Při vyšších prahových hodnotách můžeme z dat odstranit nepotřebné struktury, na druhou stranu nejsou výsledné projekce natolik vyhlazené, jako tomu je právě u nižších prahových hodnot. Pomocí nižších prahových hodnot jsme zase schopni vykreslit i struktury, které by jinak byly metodou MIP přehlédnuty.

Z hlediska hodnocení časové náročnosti jsou metody MIP a AIP daleko rychlejší při generování projekcí oproti metodě CVP. Dalším získaným poznatkem z testování algoritmu je výrazné navýšení jak časové, tak výpočetní náročnosti při nárůstu rozlišení vstupních dat.

Jako návrh na rozšíření algoritmu se nabízí doplnění o možnosti pre-processingu a post-processingu dat. Přednostně se jedná o implementování metody samotné segmentace dat. Výstupní výsledky by tak byly více přesnější a zároveň by také došlo k urychlení procesu generování projekcí. Z čistě programovacího pohledu by zcela jistě došlo k vylepšení algoritmu cíleným využitím paralelních výpočtů, které lze realizovat pomocí vícejádrových procesorů výpočetního zařízení. Výsledný algoritmus by tak byl schopný efektivnějšího využití výkonu zařízení a proces vytváření 2D projekcí by se výrazně urychlil.

Literatura

[1] BRETT, Matthew, et al. *nipy/nibabel: 3.2.1* [online]. Zenodo, 2020, [cit. 2022-05-23]. DOI: 10.5281/zenodo.4295521

[2] CASPER, da Costa-Luis, et al. *Tqdm: A fast, Extensible Progress Bar for Python and CLI* [online]. Zenodo, 2022 [cit. 2022-05-23]. DOI: 10.5281/zenodo.6412640

[3] CLINE, Harvey Ellis a Siegwalt LUDKE. *Fast method of creating 3D surfaces by "Stretching cubes"*. 1997. USA. US6115048A. Uděleno 5. 9. 2000. Zapsáno 21. 1. 1997.

[4] DALRYMPLE, Neal C., Srinivasa R. PRASAD, Michael W. FRECKLETON a Kedar N. CHINTAPALLI. Introduction to the Language of Three-dimensional Imaging with Multidetector CT. *RadioGraphics*. 2005, **25**(5), 1409-1428. ISSN 0271-5333. DOI: 10.1148/rg.255055044

[5] DREBIN, Robert A., Loren CARPENTER a Pat HANRAHAN. Volume Rendering. *ACM SIGGRAPH Computer Graphics*. 1988, **22**(4), 65-74. ISSN 0097-8930. DOI: 10.1145/378456.378484

[6] DUARTE, Marcos. *Detecta: A Python module to detect events in data* [online]. Zenodo, 2021 [cit. 2022-05-23]. DOI: 10.5281/zenodo.4598962

[7] FISHMAN, Elliot K., Derek R. NEY, David G. HEATH, Frank M. CORL, Karen M. HORTON a Pamela T. JOHNSON. Volume Rendering versus Maximum Intensity Projection in CT Angiography: What Works Best, When, and Why. *RadioGraphics*. 2006, **26**(3), 905-922. ISSN 0271-5333. DOI: 10.1148/rg.263055186

[8] GOSHTASBY, Ardeshir A. *2-D and 3-D Image Registration for Medical, Remote Sensing, and Industrial Applications*. Hoboken: Wiley-Interscience, 2005, 270 s. ISBN 978-0471724278.

[9] GROLL, Jakub. *Dvojúrovňové vykreslování volumetrických dat*. Praha, 2017. Diplomová práce. České vysoké učení technické v Praze, Katedra počítačové grafiky a interakce. Vedoucí práce Ladislav Čmolík.

[10] HLAVÁČ, Václav a Milan ŠONKA. *Počítačové vidění*. Praha: Grada, 1992, 252 s. ISBN 80-85424-67-3.

[11] HUNTER, J. D., et al. *matplotlib/matplotlib: REL: v3.5.2* [online]. Zenodo, 2022, [cit. 2022-05-23]. DOI: 10.5281/zenodo.6513224

[12] JAN, Jiří. *Medical image processing, reconstruction and restoration: Concepts and Methods*. Boca Raton: Taylor & Francis, 2005, 54 s. ISBN 0-8247-5849-8.

[13] KAZÍK, Jiří. *Vizualizace objemových dat pomocí volume renderingu*. Brno, 2007. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav počítačové grafiky a multimédií. Vedoucí práce Přemysl Kršek.

[14] KRUEGER, Wolfgang. Volume rendering and data feature enhancement. *ACM SIGGRAPH Computer Graphics*. 1990, **24**(5), 21-26. ISSN 0097-8930. DOI: 10.1145/99308.99312

[15] LACROUTE, Philippe a Marc LEVOY. 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 - SIGGRAPH '94*. New York, USA: ACM Press, 1994, 451-458. ISBN 0-89791-667-0. DOI: 10.1145/192161.192283

[16] LORENSEN, William E. a Harvey E. CLINE. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*. New York: ACM Press, 1987, **21**(4), 163-169. ISBN 0-89791-227-6. DOI: 10.1145/37401.37422

[17] LOWEKAMP, Bradley C., David T. CHEN, Luis IBÁÑEZ a Daniel BLEZEK. The Design of SimpleITK. *Frontiers in Neuroinformatics* [online]. 2013, **7** [cit. 2022-05-23]. ISSN 1662-5196. DOI: 10.3389/fninf.2013.00045

[18] MAŘÍK, Robert. *Lokální extrémy, průběh funkce*. Matematika (nejen) pro krajináře a nábytkáře [online]. Mendelova univerzita v Brně: Lesnická a dřevařská fakulta, c2007-2012 [cit. 2022-05-23]. Dostupné z: https://user.mendelu.cz/marik/mat-web/mat-webse6.html#x11-80006

[19] MASON, D. L., et al. *Pydicom: An open source DICOM library* [online]. Zenodo, 2021 [cit. 2022-05-23]. DOI: 10.5281/zenodo.6394735

[20] MIKULKA, Jan. *Segmentační metody ve zpracování biomedicínských obrazů*. Brno, 2011. Disertace. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedoucí práce Eva Gescheidtová.

[21] MORNSTEIN, Vojtěch, Vladan BERNARD, Marek DOSTÁL, Ivo HRAZDIRA, Erik STAFFA, Jaromír ŠRÁMEK a Daniel VLK. *Lékařská fyzika a biofyzika*. Brno: Masarykova univerzita, 2018, 339 s. ISBN 978-80-210-8984-6.

[22] MUCHOVÁ, Tereza. *Detekce a modelování zájmových objektů z MR mozkových dat*. Ostrava, 2018. Bakalářská práce. Vysoká škola báňská – Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky. Vedoucí práce Jan Kubíček.

[23] PHAM, Dzung L., Chenyang XU a Jerry L. PRINCE. Current Methods in Medical Image Segmentation. *Annual Review of Biomedical Engineering*. 2000, **2**(1), 315-337. ISSN 1523-9829. DOI: 10.1146/annurev.bioeng.2.1.315

[24] PROKOP, Mathias, Hoen Oh SHIN, Ansgar SCHANZ a Cornelia M. SCHAEFER-PROKOP. Use of Maximum Intensity Projections in CT Angiography: A Basic Review. *Radiographics*. 1997, **17**(2), 433-451. ISSN 0271-5333. DOI: 10.1148/radiographics.17.2.9084083

[25] SATO, Yoshinobu, Nobuyuki SHIRAGA, Shin NAKAJIMA, Shinichi TAMURA a Ron KIKINIS. Local Maximum Intensity Projection (LMIP): A New Rendering Method for Vascular Visualization. *Journal of Computer Assisted Tomography*. 1998, **22**(6), 912-917. DOI: 10.1097/00004728-199811000-00014

[26] SHARMA, Neeraj, Amit K. RAY, K. K. SHUKLA, Shiru SHARMA, Satyajit PRADHAN, Arvind SRIVASTVA a Lalit M. AGGARWAL. Automated medical image segmentation techniques. *Journal of Medical Physics*. 2010, **35**(1). ISSN 0971-6203. DOI: 10.4103/0971-6203.58777

[27] SHIRLEY, Peter a Allan TUCHMAN. A polygonal approximation to direct scalar volume rendering. *ACM SIGGRAPH Computer Graphics*. 1990, **24**(5), 63-70. ISSN 0097-8930. DOI: 10.1145/99308.99322

[28] SOUČEK, Tomáš. *Segmentace CT snímků a tvorba 3D dat pro CAD technologie*. Liberec, 2017. Bakalářská práce. Technická univerzita v Liberci, Fakulta zdravotnických studií. Vedoucí práce Jan Koprnický.

[29] SOUČEK, Tomáš. *Segmentace MR obrazu*. Liberec, 2019. Diplomová práce. Technická univerzita v Liberci, Fakulta zdravotnických studií. Vedoucí práce Daniel Jirák.

[30] ŠPANĚL, Michal a Vítězslav BERAN. *Obrazové segmentační techniky: Přehled existujících metod*. Vysoké učení technické v Brně: Fakulta informačních technologií [online]. Brno: Vysoké učení technické v Brně, 2005 [cit. 2021-12-24]. Dostupné z: http://www.fit.vutbr.cz/ spanel/segmentace/

[31] TIŠNOVSKÝ, Pavel. *Zobrazení objemových dat v POV-Rayi*. Root.cz [online]. Internet Info, c1998-2021, 14. 10. 2008 [cit. 2021-12-24]. Dostupné z: https://www.root.cz/clanky/zobrazeni-objemovych-dat-v-pov-rayi/

[32] TIŠNOVSKÝ, Pavel. *Jupyter Notebook – nástroj pro programátory, výzkumníky i lektory*. Root.cz [online]. Internet Info, c1998-2021, 21. 4. 2020 [cit. 2022-05-23]. Dostupné z: https://www.root.cz/clanky/jupyter-notebook-nastroj-pro-programatory-vyzkumniky-i-lektory/

[33] UHER, Vojtěch. *Vizualizace volumetrických dat*. Ostrava, 2013. Diplomová práce. Vysoká škola báňská – Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra informatiky. Vedoucí práce Tomáš Fabián.

[34] VAN GELDER, Allen a Kwansik KIM. Direct volume rendering with shading via three-dimensional textures. In: *Proceedings of 1996 Symposium on Volume Visualization*. Santa Cruz (California): ACM, 1996, 23-30. ISBN 0-89791-865-7. DOI: 10.1109/SVV.1996.558039

[35] YANIV, Ziv, Bradley C. LOWEKAMP, Hans J. JOHNSON a Richard BEARE. SimpleITK Image-Analysis Notebooks: a Collaborative Environment for Education and Reproducible Research. *Journal of Digital Imaging*. 2018, **31**(3), 290-303 [cit. 2022-05-23]. ISSN 0897-1889. DOI: 10.1007/s10278-017-0037-8

[36] ŽÁRA, Jiří, Bedřich BENEŠ, Jiří SOCHOR a Petr FELKEL. *Moderní počítačová grafika*. 2. přepracované a rozšířené vydání. Brno: Computer Press, 2005, 609 s. ISBN 80-251-0454-0.

[37] *Pixels, Voxels, and Signal Intensity*. Aibolita.com [online]. [cit. 2021-12-01]. Dostupné z: https://aibolita.com/cancer/49656-pixels-voxels-and-signal-intensity.html

[38] *Current Edition*. DICOM Standard [online]. Arlington: Medical Imaging & Technology Alliance (MITA) [cit. 2022-05-23]. Dostupné z: https://www.dicomstandard.org/current/

[39] *DICOM Standart Browser*. Innolitics [online]. Austin: Innolitics, c2016–2022 [cit. 2022-05-23]. Dostupné z: https://dicom.innolitics.com/ciods

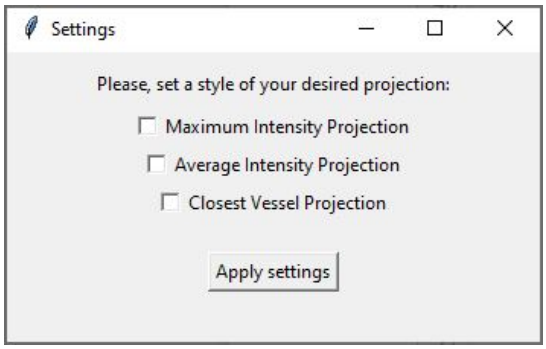
[40] *The Huygens MIP Renderer: Obtain 3D to 2D projections of the highest intensities in an object*. Scientific Volume Imaging: Deconvolution - Visualization - Analysis [online]. Hilversum: Scientific Volume Imaging B.V. [cit. 2021-12-29]. Dostupné z: https://svi.nl/Huygens-MIP-Renderer

Seznam symbolů a zkratek

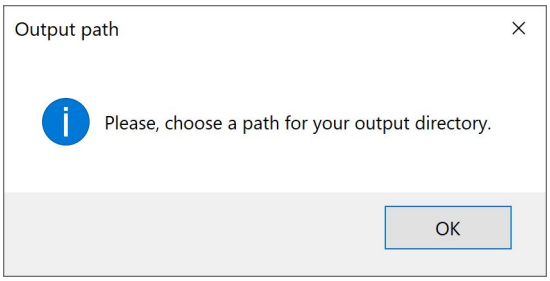
AIP	projekce průměrné intenzity - Average Intensity Projection
CPU	centrální procesorová jednotka - Central Processing Unit
CT	výpočetní tomografie - Computed Tomography
CVP	projekce nejbližších cév - Closest Vessel Projection
DOF	stupně svobody - Degrees of Freedom
DVR	přímé zobrazení volumetrických dat - Direct Volume Rendering
FIFO	první dovnitř, první ven - First In, First Out
GUI	grafické uživatelské prostředí - Graphic User Interface
LMIP	projekce lokální maximální intenzity - Local Maximum Intensity Projection
MIP	projekce maximální intenzity - Maximum Intensity Projection
MR	magnetická rezonance
NURBS	Non-uniform rational basis spline
RAM	operační paměť - Random Access Memory
RC	vrhání paprsku - Ray Casting
REPL	v reálné smyčce vyhodnocené příkazy- Real-Eval-Print-Loop
RT	sledování paprsku - Ray Tracing

Seznam příloh

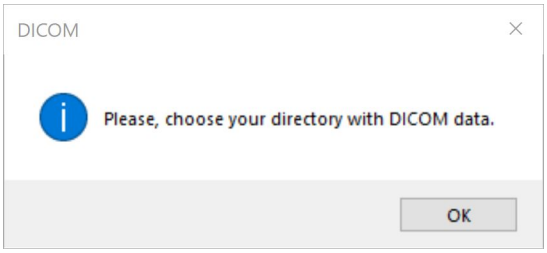
A Ukázky widgetů programu pro interakci s uživatelem



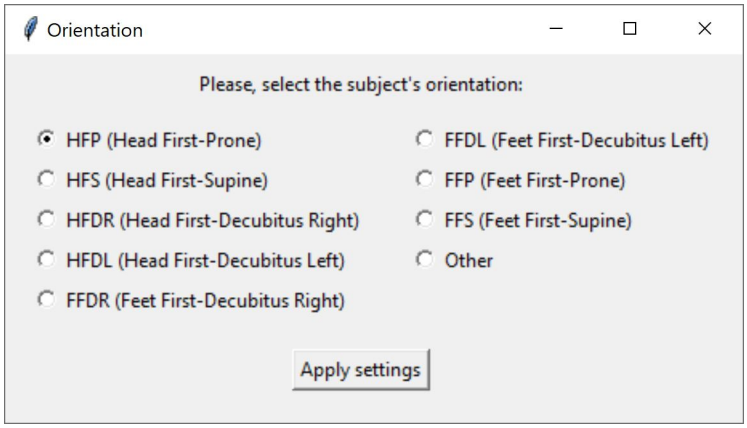
Obr. A.1: Nastavení požadovaného typu projekce



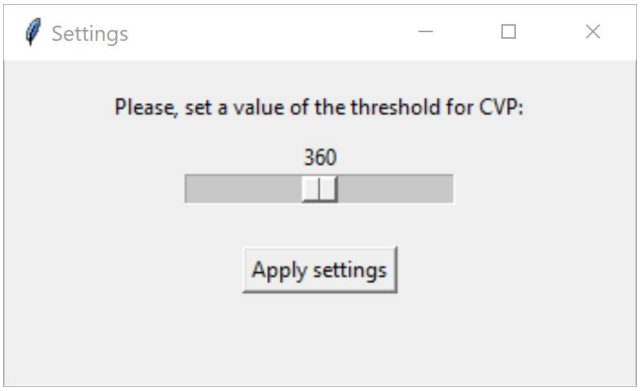
Obr. A.2: Upozornění na výběr výstupní cesty



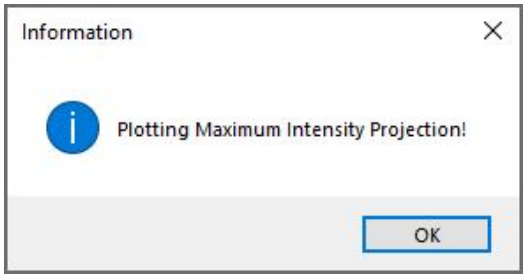
Obr. A.3: Upozornění na výběr vstupního datasetu



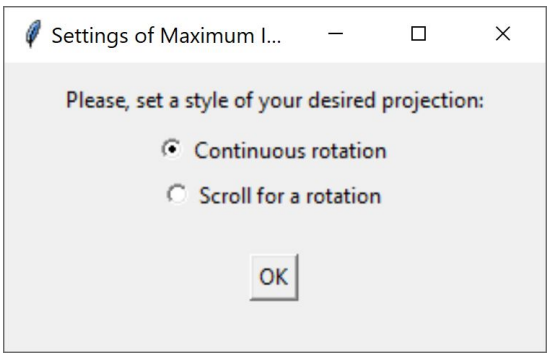
Obr. A.4: Výběr orientace subjektu



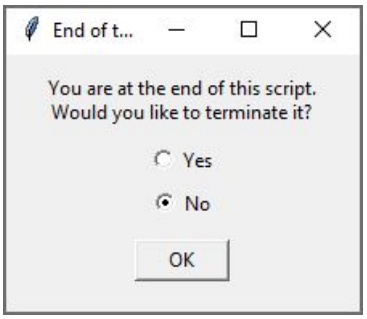
Obr. A.5: Nastavení prahové hodnoty u metody Closest Vessel Projection



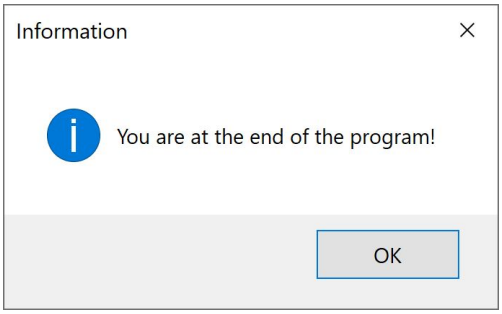
Obr. A.6: Informativní widget upozorňující na typ zobrazované projekce



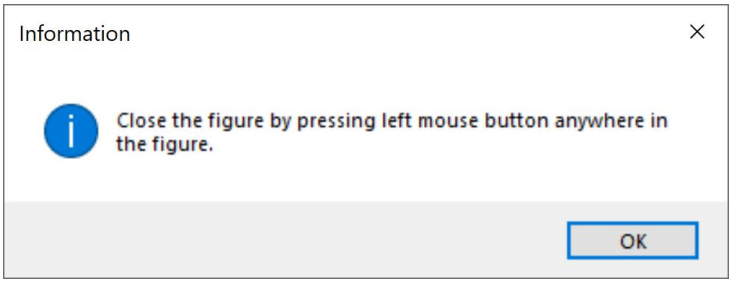
Obr. A.7: Výběr typu zobrazení dané projekce



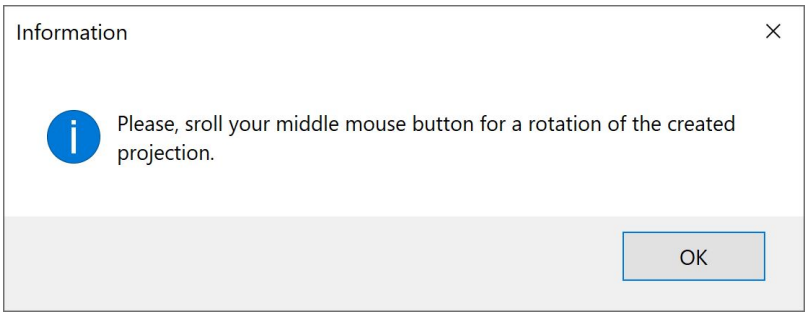
Obr. A.8: Nabídka ukončení skriptu



Obr. A.9: Informativní widget oznamující ukončení skriptu

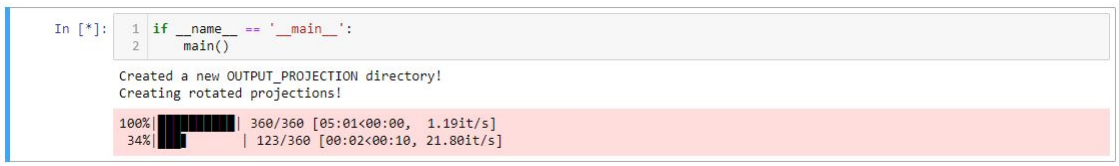


Obr. A.10: Informativní widget upozorňující na způsob ukončení vykreslování grafu



Obr. A.11: Informativní widget upozorňující na způsob ovládání vykreslování grafu

Obr. A.12: Výpis v konzoli informující o množství vytvořených 2D projekcí



Obr. A.13: Výpis v konzoli informující o vytváření fronty z 2D projekcí

B Obsah elektronické přílohy

- `/` ..... kořenový adresář přiloženého archivu
  - `vstupni_dataset` ..... ukázkový vstupní dataset
    - `Head.tar`
  - `vystup_ukazka` ..... vygenerovný výstup sloužící pro ukázku
    - `head_cvp` ..... Metoda CVP aplikovaná na dataset Head
      - `CVP_Head_000.nii.gz`
      - `CVP_Head_001.nii.gz`
      - ⋮
      - `CVP_Head_359.nii.gz`
    - `head_mip` ..... Metoda MIP aplikovaná na dataset Head
      - `MIP_Head_000.nii.gz`
      - `MIP_Head_001.nii.gz`
      - ⋮
      - `MIP_Head_359.nii.gz`
  - `michaela_viktorinova_skript.ipynb` ..... vytvořený skript
  - `readme.txt` ..... soubor s popisem běhu přiloženého skriptu