VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Využití rozšířené reality pro vizualizaci 3D medicínských obrazových dat

Using Augmented Reality to Visualize 3D Medical Images

Bc. Lukáš Hojdysz

Diplomová práce

Vedoucí práce: Mgr. Ing. Michal Krumnikl, Ph.D.

Ostrava, 2023

VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Č.j.VSB/22/078939
vsboee2202602D

Zadání diplomové práce

Student: **Bc. Lukáš Hojdysz**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Využití rozšířené reality pro vizualizaci 3D medicínských obrazových dat
Using Augmented Reality to Visualize 3D Medical Images

Jazyk vypracování: čeština

Zásady pro vypracování:

Vizualizace 3D medicínských dat je oblastí počítačové grafiky s extrémně vysokou výpočetní náročností používaných algoritmů při současném požadavku na vysokou přesnost výpočtů. Jako příklad lze uvést zobrazení objektu vyjádřeného řádově stovkami elementů rychlostí několikrát za sekundu, přičemž se předpokládá přesné umístění ve scéně vykreslované pomocí zařízení rozšířené reality (AR).

1. Proveďte analýzu existujících řešení pro vizualizaci 3D medicínských dat v prostředí AR.
2. Vyberte a popište vhodné hardwarové prostředky pro rozšířenou realitu.
3. Vyberte a popište nástroje pro vykreslování 3D medicínských dat a jejich vhodnost pro použití v oblasti AR.
4. Navrhněte aplikaci umožňující zobrazovat 3D medicínská data v prostředí rozšířené reality.
5. Navržené řešení implementujte a vyhodnoťte přesnost lokalizace vykreslovaného objektu.
6. Proveďte testy se zvoleným hardwarem pro rozšířenou realitu.

Seznam doporučené odborné literatury:

[1] AVERSA Davide, DICKINSON Chris. Unity Game Optimization: Enhance and extend the performance of all aspects of your Unity games, 3rd Edition. Packt Publishing. 2019. ISBN 978-1838556518
[2] BORROMEO Nicolas Alejandro. Hands-On Unity 2020 Game Development: Build, customize, and optimize professional games using Unity 2020 and C#. Packt Publishing. 2020. ISBN 978-1838642006
[3] LINOWES Jonathan, BABILINSKI Krystian. Augmented reality for developers: Build practical augmented reality applications with unity, ARCore, ARKit, and Vuforia. Packt Publishing Ltd, 2017. ISBN 978-1787286436

17. listopadu 2172/15
708 00 Ostrava-Poruba
Česká republika

spojovatelka: +420 597 321 111
epodatelna: epodatelna@vsb.cz
ID datové schránky: d3kj88v

IČ: 61989100
DIČ: CZ61989100

email: studijni.fei@vsb.cz
www.fei.vsb.cz

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Datum zadání: 01.09.2022

Datum odevzdání: 30.04.2023

Garant studijního oboru: prof. RNDr. Václav Snášel, CSc.

V IS EDISON zadáno: 07.11.2022 11:59:22

Abstrakt

Tato diplomová práce se zabývá využitím rozšířené a virtuální reality v medicínském odvětví k prostorovému zobrazení objemových dat z výpočetní tomografie (CT). Práce se zabývá jak přímou zobrazovací metodou (volume rendering), při které dochází k využití původních lékařských dat pro vykreslování (typicky DICOM), tak i nepřímou metodou při které jsou využity klasické modely. Textová část práce obsahuje část teoretickou a část praktickou, ve které je pak popsán postup vývoje softwaru. Práce vznikla ve spolupráci s Fakultní nemocnicí Ostrava a software je primárně určen pro jejich potřeby. Předpokládá se, že v budoucnu bude software v nemocnici sloužit jako alternativní možnost zobrazení dat z výpočetní tomografie.

Klíčová slova

DICOM; NRRD; volumetrické vykreslování; raymarching; vykreslování modelů; Hololens 2; Quest 2; smíšená realita; virtuální realita; Unity; Blender; FNO; grafické shadery

Abstract

This master's thesis focuses on the use of augmented and virtual reality in the medical field for spatial visualization of volumetric data from computed tomography (CT). The work investigates both direct visualization method (volume rendering), which utilizes the original medical data for rendering (typically DICOM), and indirect method using traditional models. The text portion of the thesis consists of a theoretical section and a practical section, in which the software development process is described. The project was created in collaboration with Fakultni nemocnice Ostrava, and the software is primarily intended for their needs. It is expected that in the future, the software will serve as an alternative option for displaying data from computed tomography in the hospital.

Keywords

DICOM; NRRD; volume rendering; raymarching; surface model rendering; Hololens 2; Quest 2; mixed reality; virtual reality; Unity; Blender; FNO; graphics shaders

Poděkování

Děkuji panu Mgr. Ing. Michalu Krumniklovi, Ph.D. za spolupráci, odborné vedení a cenné rady při zpracování této diplomové práce. Poděkování rovněž patří panu Ing. Martinu Němcovi, Ph.D., který měl v průběhu vývoje praktické poznámky pro vylepšení aplikace. V neposlední řadě bych chtěl poděkovat Fakultní nemocnici Ostrava a především jejímu týmu chirurgů za vzájemnou spolupráci při vývoji této práce.

Obsah

Seznam použitých zkratek a symbolů

CT	–	Computed tomography (výpočetní tomografie)
MRI	–	Magnetic resonance imaging (magnetická rezonance)
HMD	–	Head-mounted display (displej umístěný na hlavě)
AR	–	Augmented reality (rozšířená realita)
VR	–	Virtual reality (virtuální realita)
MR	–	Mixed reality (smíšená realita)
XR	–	Extended reality (souhrnný termín, který v sobě zahrnuje AR/-VR/MR)
FNO	–	Fakultní nemocnice Ostrava
MIP	–	Maximum intensity projection (technika zobrazující největší intenzity v cestě paprsku
DVR	–	Direct volume rendering (technika umožnující skládání hustotních/intenzitních hodnot v cestě paprsku s aplikovanou transfer funkcí
TF	–	Transfer funkce (funkce přiřazující barvu a průhlednost objemovým datům na základě jejich hustoty/intenzity a případně dalších parametrů)
MRTK	–	Mixed Reality Toolkit (knihovna poskytující ovládací prvky pro MR/VR)
LTS	–	Long-term support (verze s dlouhodobou podporou)
DICOM	–	Digital Imaging and Communications in Medicine (standard pro distribuci a skladování medicínských dat)
NRRD	–	Nearly raw raster data (univerzální standard pro distribuci a skladování převážně objemových dat)
NIfTI	–	Neuroimaging Informatics Technology Initiative (standard pro distribuci a skladování převážně neurologických zobrazovacích dat)
GPU	–	Graphics processing unit (grafický procesor)
CPU	–	Central processing unit (centrální procesorová jednotka)
FPS	–	Frames per second (snímky za sekundu)

UWP	–	Universal Windows Platform
JIT	–	Just-in-time compilation
AOT	–	Ahead-of-time compilation
WPF	–	Windows Presentation Foundation
FOV	–	Field of view (zorné pole)

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

S posledním trendem vývoje moderních technologií hlavně v oblasti rozšířené a virtuální reality je snaha tyto technologie využít i v medicínském odvětví, které by mohlo těžit z možnosti zobrazování reálných dat získaných z výpočetní tomografie (CT) a magnetické rezonance (MRI) v prostorové formě. Dosud bylo možné tato data zobrazovat především na monitorech, kde byla vizualizace omezena jednou dimenzí. Tradiční software byl proto navržen tak, aby osoba zkoumající tato data mohla prohlížet obraz ze všech možných stran a úhlů. Tento způsob ovládání, který se snaží eliminovat nedostatky vyplývající z absence třetí dimenze, však není ideální.

V poslední době s vývojem zařízení pro rozšířenou, popř. virtuální realitu, konkrétně v sektoru „Head mounted displays“ (HMD) přichází nové možnosti pro zobrazování těchto dat. Oba způsoby, tj. rozšířená realita (AR) a virtuální realita (VR), mají své výhody a nevýhody. Výhoda zejména v oblasti VR je taková, že prostředí obvykle uživatele neodvádí od pozornosti a uživatel se může soustředit jen na interakci s danými daty. V rozšířené realitě je pak výhoda možnost zarovnat naskenovaná data s reálnými daty v prostoru (například s pacientem nebo orgánem), lékař tak může v případě potřeby, tzv. „nahlédnout pod kůži“ pacienta. Zobrazení dat v AR nebo VR je mnohem intuitivnější než na monitoru, protože uživatel má před sebou prostorový 3D model, který si může prohlížet, jako by byl skutečný. Vzhledem k tomu, že odvětví rozšířené reality je stále v průběhu aktivního vývoje, práce s AR zařízením je značně nestabilní, a postupy se mohou s každou aktualizací AR knihoven i výrazně lišit.

Cílem této práce je vyvinout aplikaci pro zobrazení medicínských dat v 3D prostoru pomocí dvou různých metod. První metoda spočívá ve využití volume renderingu nad surovými lékařskými daty, což je metoda přímé vizualizace. Druhá metoda zahrnuje klasické vykreslování již vytvořených modelů, které vycházejí z původních dat, což je metoda nepřímé vizualizace. Lékař by měl být schopen si v obou řešeních vybírat jakou část dat si bude právě prohlížet a měl by být schopen odfiltrovat pro něj nerelevantní data. V práci bude kladen velký důraz na vhodné použití dostupných knihoven, a to hlavně z časových důvodů, ale také z důvodů zachování kompatibility s různými systémy, které bude teoreticky možné snadněji rozšířit v budoucnu.

Aplikace v této práci bude vytvářena v Unity enginu a bude cílena primárně na zařízení Hololens 2, které je v době psaní této diplomové práce pořád bráno jako jedno z nejpokročilejších zařízení v HMD AR sektoru na trhu. Vzhledem k multiplatformním výhodám v Unity bude aplikace paralelně vyvíjena i ve VR verzi, která se bude ovládat velmi podobně jako verze pro Hololens 2 až na pár drobných změn zejména v rozdílech mezi ovládáním pomocí snímání rukou (hand tracking) na Hololens 2 a ovládáním pomocí ovladačů ve VR.

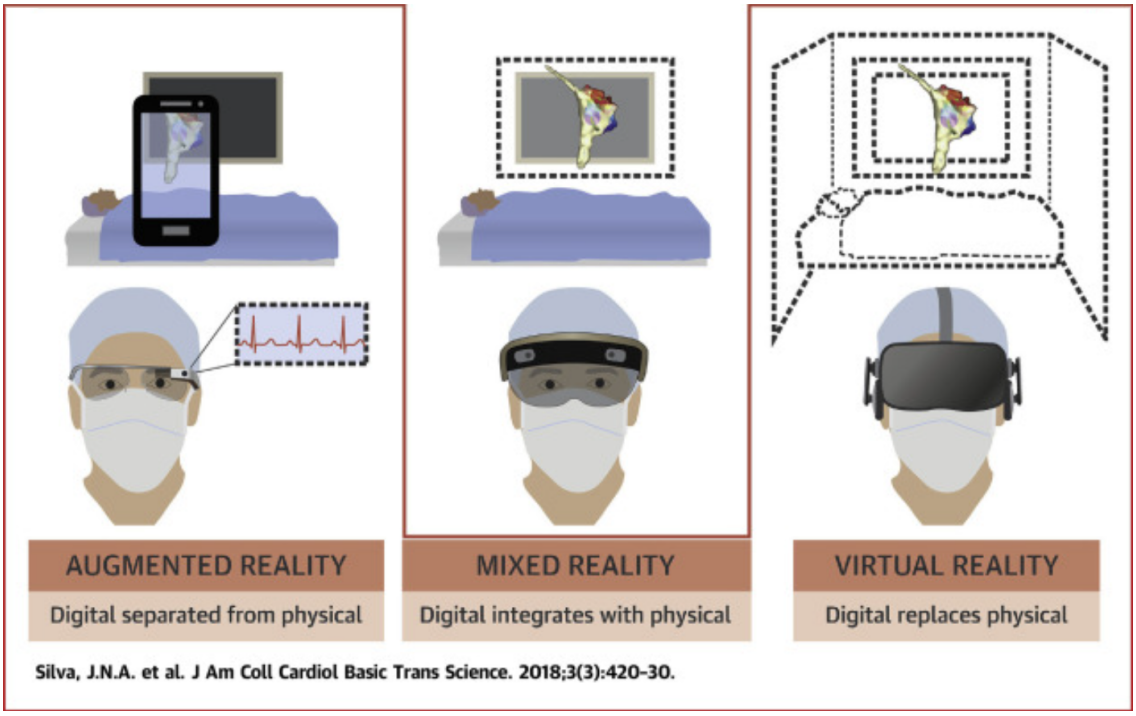
Jelikož je aplikace určena pro medicínské odvětví, je nutné dosáhnout vzájemného konsenzu mezi světem programátorů a lékařů, kde ovládání těchto aplikací pro rozšířenou a virtuální realitu musí být natolik intuitivní, že lékaři budou schopni tyto aplikace ovládat a vhodně je využít ke svému prospěchu. Vzájemná komunikace je tedy naprosto klíčová k dosažení aplikace, která má reálný smysl.

Tato práce vzniká ve spolupráci s Fakultní nemocnicí Ostrava (FNO) a její chirurgickou klinikou, se kterou jsme byli v době vývoje v aktivním kontaktu a dostávali jsme cennou zpětnou vazbu.

Kapitola 2

Zkoumání dosavadních vizualizačních řešení v oboru

Pro úplnost je třeba na úvod zmínit, že tato práce se bude zaměřovat na takzvanou „Extended reality“ (XR), která zahrnuje „Virtual Reality“ (VR), „Augmented reality“ (AR) a „Mixed reality“ (MR). Většina pozornosti v této práci bude věnována právě MR, což lze chápat jako vylepšenou rozšířenou realitu, která místo pouhého překrytí reálných prvků umožňuje s hologramy i manipulovat. Pro přehled jsou rozdíly znázorněny na obrázku 2.1.



Obrázek 2.1: Odlišnosti v XR (převzato z [1])

Vzhledem k tomu, že odvětví vizualizace medicínských dat pomocí XR je stále relativně nové, existuje v současnosti jen málo aplikací zabývajících se tímto tématem. Avšak v období 2018-2023 vznikla celá řada studií a vědeckých článků, které se těmito nebo velmi blízkými tématy zabývají. Tato sekce obsahuje výběr článků a studií, které byly z hlediska přínosu v dané problematice za poslední dobu nejvýznamnější. Nejsou zde uvedeny všechny články z oboru, i když zaměřením mohly být podobné, protože se informace často opakovaly ([1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12]). Kromě článků a studií existuje také řada dalších zdrojů pokrývajících danou tématiku, o kterých se poté píše v sekci 2.2.

2.1 Klinické články a studie s využitím rozšířené reality

Pro úvodní orientaci v problematice rozšířené reality v prostředí nemocnic lze doporučit [1], kde je shrnut stav rozvoje technologie do roku 2018. Dokument vysvětluje základní principy, časté problémy a nejznámější firmy působící v oblasti nemocniční rozšířené reality .Článek [2] pak mapuje přínosné studie v daném odvětví vzniklých v období 2012-2017.

O výhodách využití AR pro vizualizaci CT snímků při pitvách se zabýval článek [3]. Autoři článku prezentují řešení, které umí zobrazovat jednotlivé CT řezy v zařízení. Chirurg se tedy nemusí obracet na monitor a může mít zvolené řezy pořád před sebou. Ukázka tohoto přístupu je vyobrazena na obrázku 2.2.

Obrázek 2.2: Zobrazení CT snímků v AR (převzato z [3])

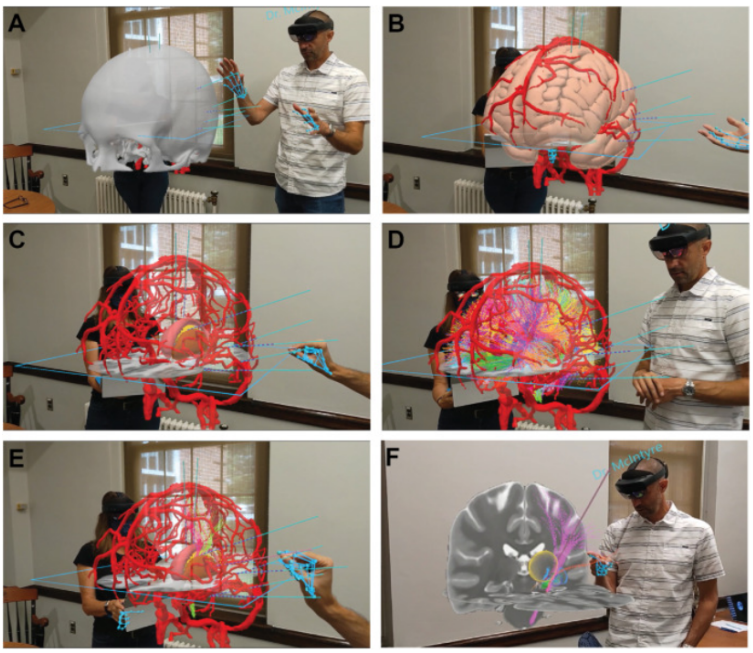
Jako nespornou výhodu autoři zmiňují, že nedochází ke zbytečné kontaminaci, neboť chirurg může Hololens aplikaci ovládat rukama v prostoru a ničeho se nedotýká. Autoři však zmiňují také limitace práce s AR, jako třeba dlouhá doba nahrání potřebných dat do zařízení, omezené zorné pole (field of view - FOV) a váhu zařízení, která může po delším čase způsobovat bolesti hlavy.

Z oblasti neurochirurgie, pochází článek [4], který získal prostor na Microsoft konferenci. Článek také pokrývá tato zpráva [13]. Text publikovaného článku se zabývá sedmiletým působením autorů v daném odvětví. Autoři zdůrazňují tyto hlavní body svého řešení:

- holografická vizualizace a interaktivní selekce relevantních anatomických struktur,

- využití koordinačního systému používaném v klinické praxi,
- možnosti umístění či přemístění intrakraniálních eletrod a simulace „axonal pathway activation“ v holografickém modelu pacienta,
- možnost zároveň prohlížet ta samá data více účastníky, kteří se navzájem vidí a mohou komunikovat prostřednictvím VoIP.

Autoři nicméně uvádějí, že aplikace zatím není ve stavu vhodném pro použití v klinickém prostředí a slouží převážně akademickým potřebám. Aplikaci lze vyzkoušet na následujících odkazech (odkaz1, odkaz2). Metoda zobrazení, kterou využívají, je vidět na obrázku 2.3. V článku je rovněž přiloženo demonstrativní video ukazující uživatelskou interakci.

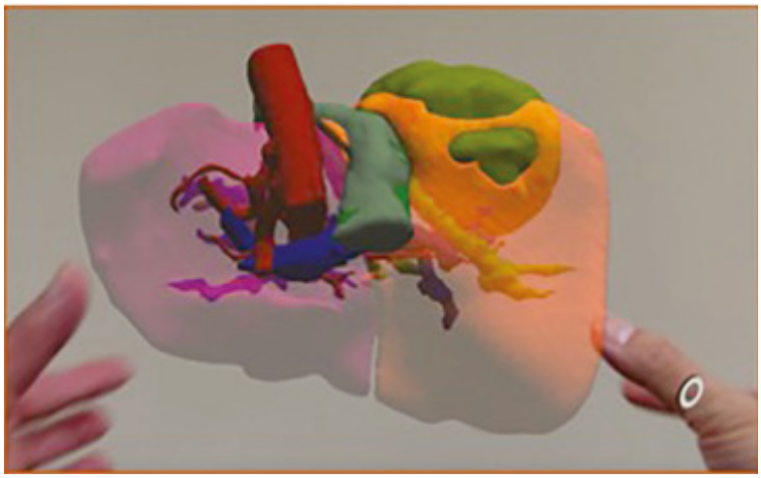


Obrázek 2.3: Vizualizace na platformě HoloSNS (převzato z [4])

Článek zabývající se tématem transplantace jater [5] uvádí ukázky s využitím Hololens 2 a vykreslováním předem vytvořených modelů. V práci autorů je velmi obohacující sekce, týkající se segmentace: „Třemi metodami objemové analýzy se segmentací pomocí 3-DRS jsou manuální, poloautomatická a automatická segmentace. Manuální segmentace byla standardní metodou objemového měření prováděná výhradně radiology, dokud se nezačal v klinické praxi používat specializovaný software. Manuální metoda se provádí pomocí dnes dostupného softwaru pro CT/MRI systémy a spočívá v precizním označení anatomických hranic na každém 2D řezu, čímž vytváří 3D strukturu. Tento postup je časově náročný a má relativně nízkou spolehlivost. Poloautomatická segmentace se ukázala jako spolehlivější možnost. Vyžaduje však uživatelský vstup pro jednu nebo více následujících činností k definici parametrů segmentace, tj. označení parenchymu, jaterních a portálních žil.

Pro konečnou zprávu je nutná určitá uživatelská zkušenost k úpravě a ověření segmentace. Ve srovnání s manuální segmentací může poloautomatická segmentace ušetřit čas a zlepšit opakovatelnost. Oproti plně automatické segmentaci může být poloautomatická segmentace přesnější a pro lékaře přijatelnější díky uživatelsky řízenému průběhu segmentačního procesu.“ převzato a přeloženo z [5]

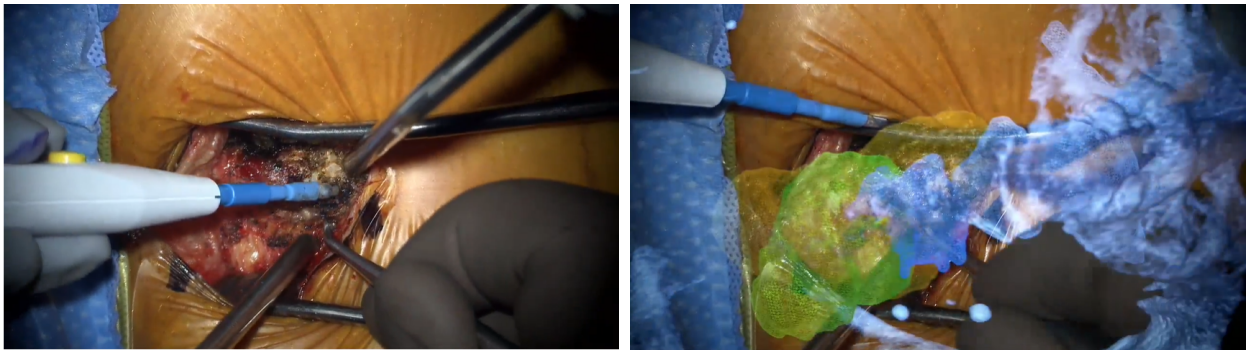
O tvorbu modelů se stará LiverVision software, který konvertuje CT/MRI data na modely. Jak autoři zmiňují v odstavci „Multicentral external validation of LiverVision“, software je nastaven na poloautomatickou segmentaci, a v softwaru jsou schopni identifikovat léze a další struktury. Zmiňují také, že vyvinutý software umožňuje prohlížení modelu více uživatelům současně. Za hlavní výhody uvádějí, že při zobrazování a manipulaci s modely nedochází ke kontaminaci. Dále také zmiňují výhody zobrazení modelů digitálně místo 3D tisku, kde digitální zobrazení je podle nich přesnější, než kdyby modely byly vytisknuty s možnými tiskovými vadami. Výhodu vidí i v ceně, kde jsou náklady pro digitální vizualizaci v porovnání s 3D tiskem minimální. Zmiňují také, že pro vykreslení modelů lze také vhodně využít barvy, kde u 3D tisku je toto problém. U zobrazení, které prezentují, lze vhodně nastavovat průhlednost jednotlivých částí, jak lze vidět na obrázku 2.4. V textu je dále zmíněno, že tato technologie se již úspěšně používá při otevřené a laparoskopické hepatektomii žijícího dárce a komplexní laparoskopické resekci jater, kde se technologie používá převážně k ověření poloh důležitých tkání a nádorů.



Obrázek 2.4: Zobrazení ze softwaru VSI-HoloMedicine (převzato z [5])

V pořadí dalším přínosným zdrojem je pak technická zpráva [6], kde autoři použili dodatečné CT k zarovnání modelu přímo na sále. Tato zpráva hodně přebírá z odborného článku [12]. Autoři zprávy používají stejnou platformu SyncAR jako autoři článku. Autoři článku však používají manuální zarovnání, zatímco ve zprávě autoři provádí zarovnání s pomocí dodatečného CT skenu, přesnost takového zarovnání je tedy velmi vysoká. Zarovnání je velmi dobře znázorněno na videu přiloženém ve zprávě. Srovnání pohledů lze také vidět na obrázku 2.5.

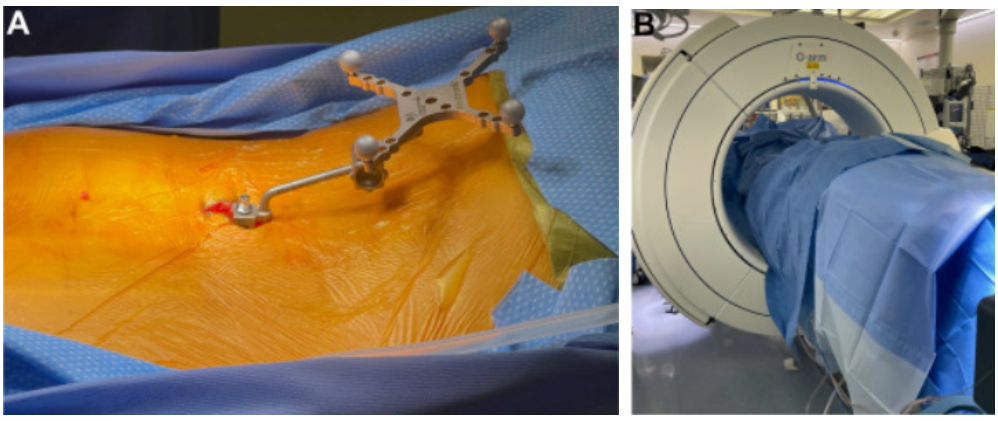
V práci autorů [6] se zkoumaly celkem tři operace, při kterých byla vizualizační technologie



(a) Pohled chirurga bez překrytí (b) Pohled chirurga s překrytím

Obrázek 2.5: Srovnání dvou pohledů (převzato z [6])

použita k přesnému zaměření orgánů a minimalizaci řezů s cílem maximálně snížit invazivnost zákroku. Autoři popisují „efektivní workflow“, úspěšnou minimalizaci invazivity a celkově dobré výsledky při práci s rozšířenou realitou (AR). Technika zarovnání modelu ve scéně, kterou využívají, je velmi zajímavá, zejména pak část zvaná „Intraoperative Navigation“. Aplikace, kterou vyvinuli, rovněž podporuje současné zobrazení AR dat pro více osob. Pacient je na operačním sále znehybněn a je mu zaveden tzv. „referenční bod“ do L5 obratle (viz obrázek 2.6 a). Následně je proveden nový CT sken, který slouží k přesnému zarovnání modelu s fyzickým, již nehybným pacientem. Během operace je pacient umístěn v CT tunelu (viz. obrázek 2.6 b)



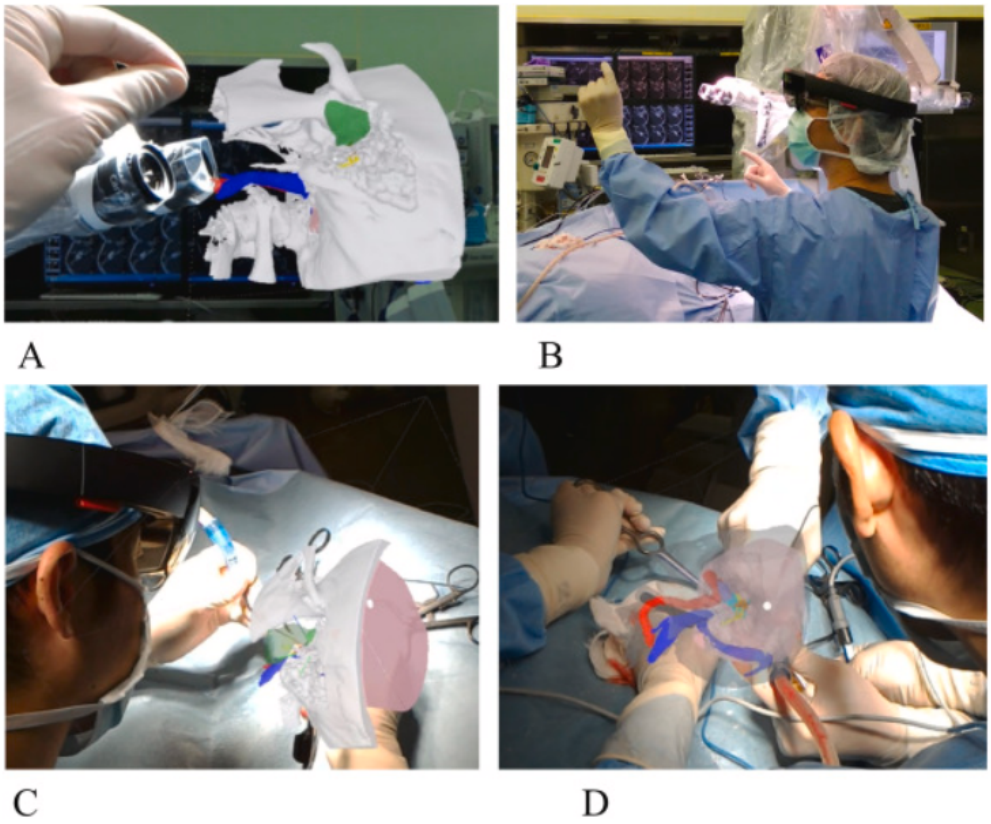
(a) Referenční svorka v L5 obratli (b) Umístění pacienta v CT tunelu

Obrázek 2.6: Zarovnání pomocí dodatečného CT skenu (převzato z [6])

Vzhledem k tomu, že chirurgický mikroskop, CT a AR zařízení jsou, jak autoři uvádějí, navzájem propojené, dochází i k augmentaci v chirurgickém mikroskopu. Přesnost navigace ověřili během operací přítomní chirurgové, kteří použili umístěný „referenční bod“ k ověření přesnosti. Úrovně průhlednosti AR překrytí a segmentaci mohli chirurgové sami během operace ovládat pomocí pedálu

k mikroskopu (microscope foot pedal) nebo překrytí mohli úplně vypnout. Všechny tři operace měly pozitivní výsledek a došlo k celkovému uzdravení pacientů. V článku je také detailněji zmíněn celkový popis stavu pacientů, specifikace nádorů a doba trvání operací. Zmínky o takovém zarovnání se objevily dříve, například ve studii [11], nicméně zde se pravděpodobně poprvé toto zarovnání úspěšně použilo při reálných operacích. Avšak jak sami autoři uznávají, je nutné brát co největší ohled na pacientovu bezpečnost, a vzhledem k tomu, že zarovnání probíhá s pomocí nového ozáření z CT, ani toto řešení není z hlediska bezpečnosti ideální.

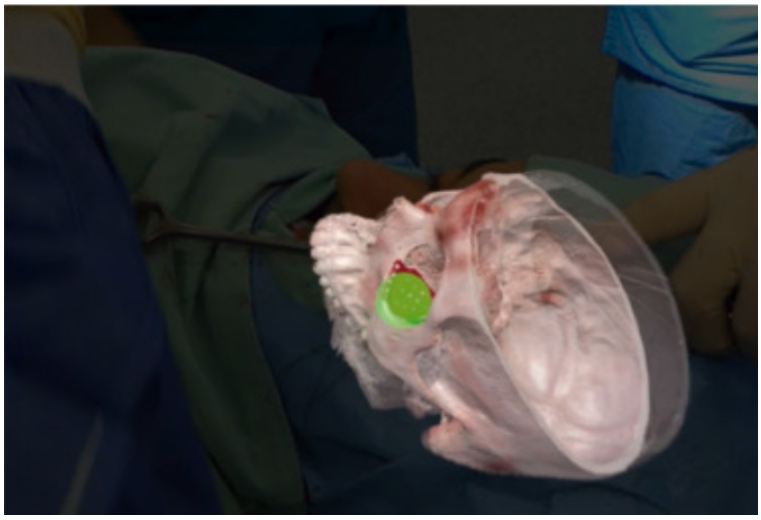
V dalším článku [7] autoři k rekonstrukci modelů používají program Slicer3D. Segmentaci provádějí pomocí prahování (tresholdingu) a k zobrazení je použit Hololens spojený přímo přes Slicer pomocí doplňku SlicerVirtualReality. Ze softwaru autoři využívají HoloeyeXR aplikaci. Zobrazení probíhá klasicky pomocí modelů. Ukázka je vidět na přiložených obrázcích 2.7.



Obrázek 2.7: Ukázka zobrazení v HoloeyesXR (převzato z [7])

V oblasti čelistní chirurgie existuje technická zpráva [8], ve které autoři provádí manuální zarovnání modelu s pacientem pomocí různých specifických bodů, jako je špička nosu, tragi a horní čelist. Zobrazení probíhá přes Hololens 1 brýle. Jak autoři uvádějí: „Krok registrace digitálních 3D modelů se skutečnou polohou hlavy pacienta stále představuje zásadní omezení při použití AR technologie na operačním sále. V našem případě jsme zarovnali digitální modely se skutečnou hlavou manuálně

s využitím brýlí Hololens 1. Registrace byla náročná a vyžadovala několik úprav zarovnání během operace.“ Dále Zmiňují „AR aplikace v medicíně a čelistní chirurgii se nachází teprve v počátečních fázích; nicméně, nese velký potenciál.“ převzato z [8]. Autoři předpokládají, že v budoucnu bude pravděpodobně AR ve standardním vybavení chirurga. Zde na obrázku 2.8 je vidět jak vypadá augmentace přes jejich D2P software od 3DSystems.



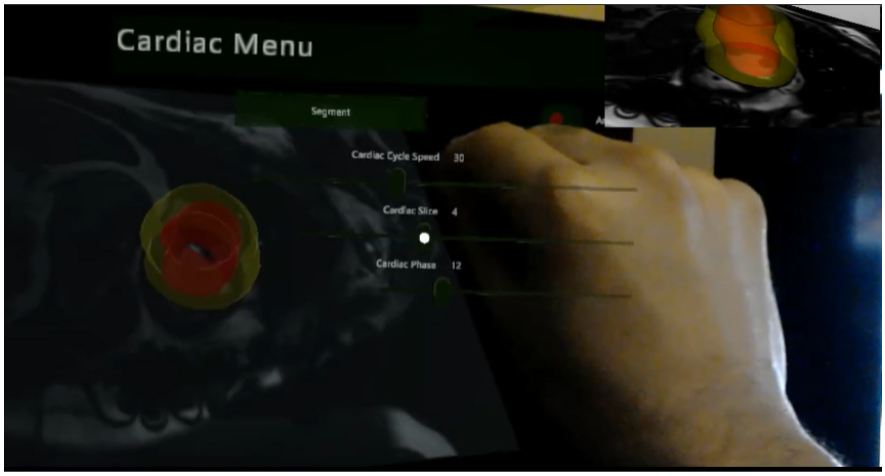
Obrázek 2.8: Zobrazení v D2P softwaru (převzato z [8])

Z oblasti mikrobiologie pochází článek [9], který zkoumá využití zobrazovacích metod v dané oblasti. Navrhované řešení ve VR nabízí alternativu ke konfokálního mikroskopu. Autoři poskytují software pro neziskové osoby zdarma na následující stránce. Aplikace obsahuje velké množství ovládacích prvků. Síťové řešení pro zobrazení dat více lidem současně funguje přes Photon, a nastavení v aplikaci lze perzistentně ukládat. Pro vizualizaci je použit volume rendering s pomocí této knihovny.

Jak autoři ve článku zmiňují, velmi významnou výhodou takového řešení je cena, kde investice do VR činí okolo 2 500$ oproti konfokálnímu mikroskopu (okolo 500 000$). Zároveň se v práci objevuje velmi důležitá zmínka : „Zjistilo se, že reprezentace založená na objemových datech byla mnohonásobně lepší než povrchový model, neboť při procesu prahování (tresholdingu) intenzit bylo ztraceno příliš mnoho informací.“ převzato a přeloženo z [9] což vyzdvihuje volumetrické řešení oproti klasickému vykreslování s modely.

Posledním zdrojem zmíněným v této sekci je článek [10], který představuje celý framework pro vizualizaci dat v AR. V práci je využita Qt knihovna pro grafické rozhraní a VTK knihovna pro vizualizaci dat. Vyvinutý software se zaměřuje převážně na vizualizaci snímků srdce, což je demonstrováno v přiloženém videu, které zobrazuje rekonstrukci tlukoucího srdce. Řešení zahrnuje také modul zabývající se automatickou segmentací srdce pomocí strojového učení. Komunikace s modulem probíhá přes TCP protokol a model neuronové sítě může být buď lokálně přítomen, nebo umístěn na vzdáleném serveru.

V práci je dále zmíněn nedostatek výpočetního výkonu u HMD zařízení, kde řešením je dvou-procesorový přístup: „Dvou procesorový přístup: hlavní počítač, na kterém běží většina výpočtů, a HMD, které primárně slouží jako zařízení pro vstupní/výstupní rozhraní.“ převzato a přeloženo z [10]. Možnou výhodou tohoto řešení je také větší výdrž baterie u HMD. Ukázka frameworku FI3D je na obrázku 2.9. Avšak řešení, které autoři prezentují, podle mých znalostí pravděpodobně využívá povrchové modely a nejedná se o přímou vizualizaci.



Obrázek 2.9: Překrytí srdce v FI3D frameworku (převzato z [10])

2.2 Ostatní zdroje a aplikace

Oblastí vzdělávání studentů se zabývá stránka Holo anatomy, kde si studenti mohou prohlížet různé části těla. Ilustrativní je sekce videí, kde lze nalézt různé druhy vizualizací. Ve všech videích jsou zobrazeny vizualizace s využitím modelů. Demo, které zpřístupnili, je dostupné k vyzkoušení v Microsoft obchodu.

V oblasti kardiologie působí společnost Echo pixel. Na svých stránkách prezentují projekty True 3D a HTG. Projekt True 3D umožnuje vytvořit holografickou projekci 3D srdce. Z dostupného videa vyplývá, že využívají přímé vykreslování a nepoužívají modely. Projekt HTG a dostupné video pak ukazuje použití softwaru na sále, kde vizualizace patrně probíhá prostřednictvím monitoru. Ukázka obou dvou projektů je na obrázku 2.10.

Další společnost působící v oblasti rozšířené reality v medicíně je SentiAR, která se pak zaměřuje na augmentaci operačního sálu s pomocí brýlí Hololens. Na dostupných videích (video 1 , video 2) vysvětlují, jak funguje jejich software CommandEP.

Další zajímavou stránkou je Medical Augmented Reality, kde lze najít více projektů. Některé z nich pracují se zařízením Hololens, jiné zkoušejí využití Xbox Kinectu pro sledování postavy a

(a) Vizualizace z True3D projektu (b) Vizualizace z HTG projektu

Obrázek 2.10: Ukázky vizualizací od společnosti EchoPixel (převzato z [14])

mapování různých tkání. Na stránkách jsou umístěna velmi zajímavá videa (video 1, video 2, video 3), ve kterých všechny ukázky využívají přímých vykreslovacích metod.

V neposlední řadě stojí za zmínku nedávný článek [15] z Nemocnice Třinec-Podlesí, kde uvádějí, že zařízení Hololens 2 již používají i při skutečných operacích srdce k vizualizaci různých dat, od modelů z CT, MRI až po lékařské zprávy a videonahrávky.

2.3 Zhodnocení

Většina zde zmíněných zdrojů a řešení využívá k vizualizaci nepřímou metodu pomocí modelů. Jak již bylo uvedeno, převod reálných dat z CT skenů na modely a vytvoření povrchů s sebou nese značnou ztrátu kvality. Autoři zde uvedených zdrojů poskytují své aplikace převážně v demo verzích. Některé aplikace, jako [9], lze podle autorů použít zdarma pro nevýdělečné osoby. Avšak žádný ze zde uvedených zdrojů neposkytuje své řešení veřejně otevřené (open-source).

Cílem této práce bude přiblížit se některým zde zmíněným zdrojům. Primární důraz bude kladen na celkovou otevřenost řešení, ze kterého bude možné v budoucnu vyjít nebo ho vylepšit. Pro vykreslování dat bude využita převážně přímá metoda vykreslování, jelikož nebyla ve zkoumaných zdrojích často zastoupena, ale i kvůli výhodám, které nabízí. Přímá metoda poskytuje nespornou výhodu práce přímo s originálními daty, což má z lékařského hlediska větší použitelnost a nedochází k žádným nepřesnostem způsobeným konvertováním dat do jiných formátů. Podle debaty s chirurgy z FNO nám byly výhody této metody potvrzeny a bylo nám sděleno, že tento způsob vizualizace je pro ně daleko přínosnější, než vizualizace pomocí modelů. Výhody a nedostatky jednotlivých metod jsou pak detailněji rozebrány v kapitole 5.

Kapitola 3

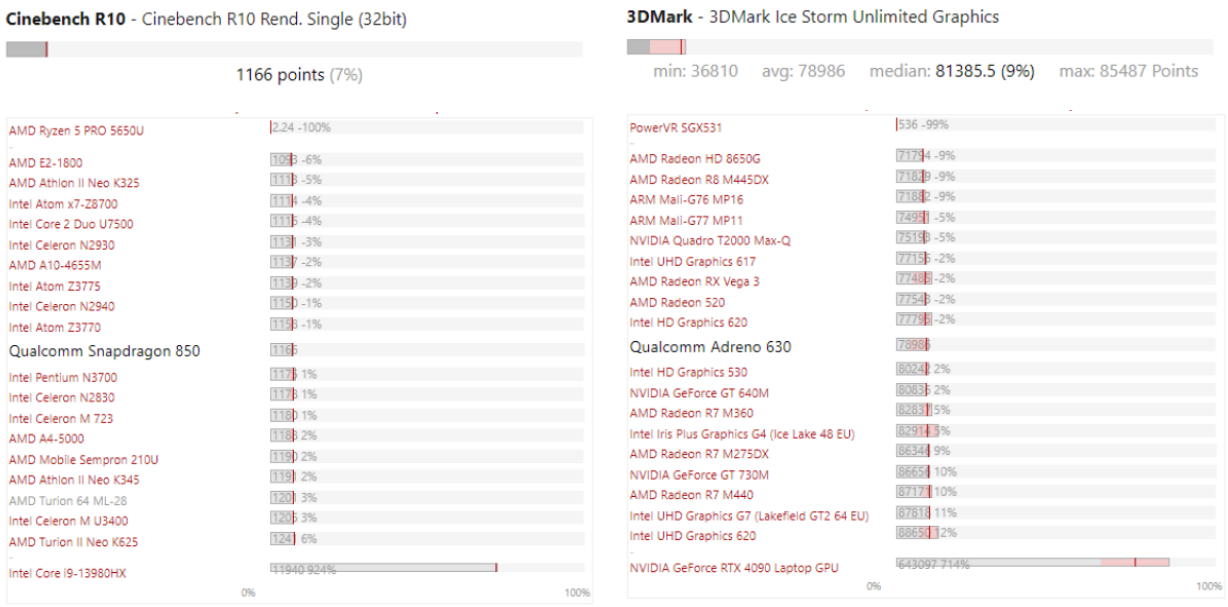
Dostupné HW prostředky pro AR

Způsob využití rozšířené reality se liší případ od případu a zahrnuje mnoho faktorů, které je třeba zohlednit. V našem případě je vhodné použít brýle pro rozšířenou realitu (HMD). Použití telefonů nebo tabletů je v našem případě nepraktické, protože se očekává co nejmenší interakce se zařízením, aby se předešlo kontaminaci. Brýle mají tu výhodu, že lékař s nimi nemusí po nasazení fyzicky manipulovat a ovládání v rozšířené realitě probíhá v prostoru pomocí detekce rukou.

V předchozí kapitole 2 bylo zřejmé, že použití Hololens a Hololens 2 je v tomto sektoru časté. Poslední dobou však dochází k čím dál častější integraci rozšířeného módu do brýlí pro virtuální realitu, což přináší nové možnosti pro výběr vhodného zařízení. Poměrně nedávný příklad je zařízení Quest Pro pro VR, které oproti předchozí verzi Quest 2 nově podporuje v rozšířeném módu barvu (Quest 2 obsahoval jen černobílý pohled z infračervených kamer). Avšak oproti zařízení Hololens 2, kde uživatel vidí své okolí přirozeně, obsahuje Quest Pro živý barevný videozáznam okolí, což může vést k problémům s orientací v prostoru, zvláště pokud není dosaženo stabilní vysoké obnovovací frekvence. Tento problém u zařízení využívající MR (kam patří i Hololens 2) nehrozí, protože uživatelé vnímají orientaci v prostoru normálně a scéna je pouze rozšířená.

Vzhledem k tomu, že Vysoká škola báňská disponuje několika zařízeními Hololens 2 a od samého začátku jsme uvažovali primárně o tomto zařízení, je v této práci využito. Zařízení Hololens 2 disponuje relativně dobrým výpočetním výkonem, díky Qualcomm Snapdragon 850 čipu a integrované grafické jednotce Adreno 630. Jednodušší úlohy, kde například dochází k vykreslení několika modelů a následné interakci s nimi, zvládne samotné zařízení bez problémů. Srovnání tohoto čipu lze vidět na obrázcích 3.1.

Je důležité vzít v úvahu, že zařízení musí zvládat nejen běh samotné aplikace, ale také obsluhu celé části systému, které zařízení disponuje. Snímání okolí, snímání rukou, snímání očí a další části ubírají podstatnou část výkonu zařízení. Běh aplikace a obsluhu všech částí systému musí zařízení zároveň zvládat s dosažením vysokého počtu snímků za sekundu. Výchozí obnovovací frekvence u zařízení Hololens 2 je 60 HZ, a vývojáři by měli na tuto hodnotu cílit během vývoje aplikace, aby



(a) Jednojádrový CPU výkon

(b) GPU výkon

Obrázek 3.1: Srovnání Qualcomm Snapdragon 850 čipu (převzato z [16], [17])

se u uživatelů předešlo nevolnostním problémům způsobeným kinetózou, která je také známa jako „Motion sickness“.

V této práci je primární větev vývoje zaměřena na přímou vizualizační metodu k zobrazení dat, která klade velmi vysoké výpočetní nároky na běh aplikace, a zařízení samo o sobě tuto vizualizaci v rozumných snímcích za sekundu nezvládne. Jak lze vidět na obrázcích 3.1, výkon mobilního čipu v porovnání se stolním CPU/GPU značně zaostává (stolní verze se nachází v posledním řádku na obou obrázcích). Výkonnostní testování bude podrobněji popsáno v kapitole 8. Stejně jako v článku [10] bude HMD zařízení v případě přímé metody primárně účinkovat v roli I/O zařízení a výpočty budou prováděny na připojeném počítači. Avšak paralelně s primární větví vývoje bude existovat i druhá větev, kde budou vizualizace probíhat vykreslováním již před-připravených modelů, a tato verze bude fungovat na samotném zařízení Hololens 2 bez nutnosti externího výpočetního výkonu. Obě dvě verze aplikace tedy půjdou srovnat. Pro zařízení Hololens 2 existuje v dokumentaci přímo sekce [18] zabývající se využitím externího výpočetního výkonu. V ideálním případě by uživatel neměl být schopen rozeznat, jestli aplikace běží externě nebo interně, avšak někteří uživatelé mohou být na mírně zvýšenou latenci citlivější, což se promítne převážně do příznaků nevolnosti při dlouhodobém používání.

Kapitola 4

Vysvětlení medicínských formátů

V oblasti medicíny se používá komplexní standard ukládání a přenosu dat zvaný Digital Imaging and Communications in Medicine (DICOM). Tento standard, vyvinutý na počátku 90. let minulého století, je kompatibilní s většinou vstupů z medicínských přístrojů, jako jsou rentgen, CT, MRI, sono a další. Ukládání dat se liší podle typu přístroje. Například u rentgenových přístrojů se obvykle používá reprezentace tvořená 2D maticí pixelů. U CT se používá reprezentace ve formě 3D matice voxelů.

DICOM formát také obsahuje různé metadata a parametry. Parametr modality například určuje typ přístroje (CT, MRI atd.), podle toho se pak určuje způsob uložení, například ztrátovým způsobem pomocí formátu JPEG, nebo některým z bezztrátových způsobů. Volba způsobu uložení může probíhat automaticky, nebo být nastavena manuálně. Při exportu dat v některých nástrojích lze třeba určit, zda mají být data uložená s kompresí, nebo bez ní. Každý DICOM snímek obsahuje dodatečné informace o pacientovi, aby nedocházelo k záměně dat. Formát lze také pro lepší spolupráci mezi institucemi anonymizovat.

Pro nás je zvláště důležitá reprezentace dat z CT skenů pacienta v DICOM formátu. Jednotlivé CT snímky ve formátu DICOM obsahují pixely s určitými hodnotami. Tyto hodnoty vychází z principu fungování CT, který zaznamenává, jak snadno rentgenové záření prochází tkáněmi pacienta. Například tkáně s vysokou hustotou, jako jsou kosti a žíly, zastaví mnohem více rentgenového záření, než tkáně s menší hustotou, jako jsou játra. Tyto hodnoty vychází z útlumových koeficientů záření a jsou přímo zaznamenány do jednotlivých pixelů na daných snímcích. Nicméně takto surová útlumová data jsou v podstatě neinterpretovatelná, neboť sken může proběhnout i s různými intenzitami. Při interpretaci je třeba data převést do správného formátu, v případě CT skenů se jedná o Hounsfieldovy jednotky [19], které představují tzv. radio-hustotu. Teoretický vzorec pro takovýto převod je vidět na rovnici 4.1.

$$HU = \frac{(\mu_{\text{material}} - \mu_{\text{water}})}{(\mu_{\text{water}})} \times 1000 \tag{4.1}$$

Hounsfieldova rovnice (převzato z [20])

Tato rovnice je však obecná a v praxi se k převodu používají dva speciální parametry dostupné z DICOM formátu, a to Rescale Slope [21] a Rescale Intercept [22]. Tyto parametry jsou v metadatech pro CT skeny povinné a umožňují převést surové útlumové hodnoty na odpovídající Hounsfieldovy jednotky radio-hustoty pomocí rovnice 4.2.

$$HU = (\text{RawPixelValue} \times \text{RescaleSlope}) + \text{RescaleIntercept} \tag{4.2}$$

Hounsfield převod využívaný v praxi [23]

Programy jako Slicer3D a některé knihovny, jako jsou ITK/SimpleITK (později použité v práci), provádějí při práci s CT snímky automaticky převod na Hounsfieldovy jednotky vyjadřující radio-hustotu (dále v textu někdy zkráceně jako hustota) dané tkáně. To je to, co si ve skutečnosti prohlížíme, nebo s čím pracujeme. Na disku jsou však data stále uložena v původním surovém formátu a k převodu dochází až při běhu programu. Díky tomu, že CT sken obsahuje typicky mnoho radiologických snímků, dá se pak z nich vytvořit prostorová voxelová mřížka s jednotlivými voxely obsahující danou hustotu. Více o principu CT skenu a o Hounsfieldových jednotkách se lze dočíst v tomto dokumentu [24].

DICOM však není jediným standardem a typem pro přenos lékařských dat. Existují i další typy formátů, jako je například populární „Nearly Raw Raster Data“ (NRRD), což je univerzální souborový formát používaný v medicíně a také v jiných odvětvích. NRRD formát je zjednodušený a generalizovaný, což umožnuje jeho použití pro různé účely. Na rozdíl od DICOMu, kde jsou data často reprezentována sekvencemi snímků, jsou všechna data v NRRD formátu uložena v jednom souboru. Více informací o NRRD formátu lze nalézt v jeho dokumentaci [25].

Dalším příkladem je pak formát „Neuroimaging Informatics Technology Initiative“ (NIfTI), který se často používá pro výstupy z MRI a je velmi oblíbený v oblasti nerozobrazování. Podobně jako NRRD, lze tento formát využít i mimo medicínské odvětví. Více informací o formátu NIfTI lze najít na stránkách [26] National Institutes of Health.

V této práci je využit jak formát DICOM, tak i NRRD formát. NRRD formát je vhodný zejména pro label mapy, které označují jednotlivé tkáně. Implementační kapitola, konkrétně podsekce 7.5.2 obsahuje více informací o exportu label map.

Kapitola 5

Způsoby vykreslování 3D medicínských dat

Pro vizualizaci lékařských dat existují primárně dva způsoby, které mají své výhody a nevýhody. Je to přímá a nepřímá vizualizace volumetrických dat.

5.1 Nepřímá vizualizace

Nepřímá vizualizace používá pro vykreslování předem vytvořené povrchové modely. Tato metoda je v mnou zkoumaných zdrojích mnohem více zastoupena než přímá vizualizace. Jedná se o klasickou metodu vykreslování předem vytvořených modelů pomocí rasterizace. Právě nízká výpočetní náročnost je často klíčovou výhodou tohoto řešení, protože nemocnice často nechtějí, nebo si nemohou dovolit pořízení externí výpočetní stanice. Řešení s povrchovými modely většinou nemá problém fungovat samostatně na zařízení jako je Hololens 2. Práce s modely se zároveň nějak zvlášť neliší od jiných oblastí, jako je například oblast vývoje her, kde využití povrchových modelů je prakticky všudypřítomné. Modelové řešení také zároveň dovoluje se prakticky vyhnout psaní nových speciálních shaderů[1], neboť ty, které jsou již standardně poskytnuty v dnešních herních enginech, většinou k vykreslování maximálně dostačují.

Nevýhodou je však, že pokud nad výstupními daty z CT nebo MRI není provedena segmentace, je prakticky nereálné toto řešení vhodně použít. Automatická segmentace, jak je zmíněno v kapitole 2 se používá, avšak tato segmentace není ideální. Pokud by se provedla prahováním, je segmentace nepřesná. Existuje i segmentace pomocí strojového učení, využitá například v [10]. Nicméně pro využití strojového učení k segmentaci je potřeba již předtrénovaný model neuronové sítě, kde trénink může zabrat velké množství času. FNO má ve spolupráci s IT 4 Inovation aktivní projekt zabývající se tréninkem modelu, který by měl být schopen rozpoznat játra a okolní orgány, nicméně tento trénink pořád probíhá a zatím není známo jakou bude mít úspěšnost.

[1] Shader je uživatelsky definovaný program, který běží na grafické kartě a určuje barvu, osvětlení, stínování a další parametry vykreslovaného objektu.

Pokud je splněn první předpoklad a je provedena segmentace, extrakce modelu z těchto dat je časově celkem náročná. V prvé řadě musí software ve kterém pracujeme, tento export podporovat. V našem případě, kde jsme pracovali převážně se softwarem Slicer 3D, je tento export možný. Export probíhá pomocí standardních algoritmů pro rekonstrukci volumetrických dat, jako je třeba Marching Cubes, nebo Flying Edges [27]. Taková renkonstrukce však vytvoří model, který je stále extrémně náročný pro vykreslení na samotném zařízení a je nutno ho ještě řádně upravit v nástrojích, jako je například Blender. Současně je nutno dát pozor, aby nedošlo k viditelné ztrátě kvality a optimalizaci provádět do rozumné míry.

Převod volumetrických dat na modely se bohužel neobejde bez ztráty na kvalitě. Existenci možných artefaktů v modelech nelze úplně zabránit. I kdyby po vytvoření modelů neproběhla optimalizace, jsou data již procesem tvorby degradovaná. Zásadním nedostatkem je také nutnost mezikroku, který vyžaduje školený personál s pokyny pro práci s modely v externích nástrojích. Vzhledem k tomu, že nemocnice komunikují výhradně prostřednictvím ustálených datových standardů jako je DICOM, práce s modely není ideální.

Pro shrnutí, výhodami nepřímé vizualizace jsou:

- relativně nízká výpočetní náročnost a přenositelnost,
- relativně jednoduchá logika aplikace,
- pro vykreslování nejsou potřeba žádné speciální shadery.

Nevýhody zahrnují:

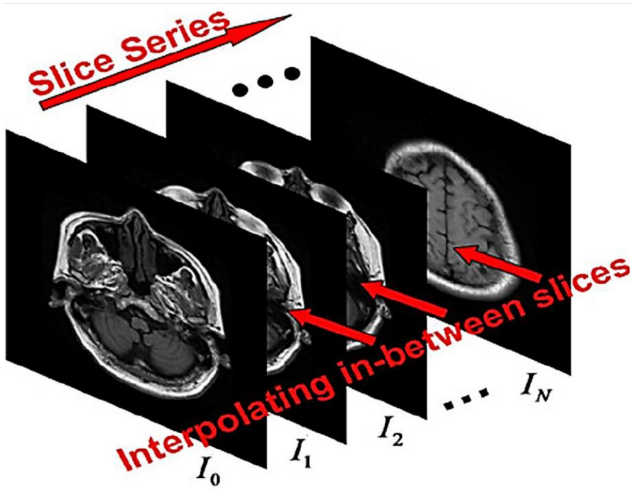
- segmentace dat je naprosto klíčová a nelze se bez ní obejít,
- vytváření a úprava modelu je časově náročná,
- model ztrácí často důležité detaily,
- nepracuje se s výchozími daty, vizualizace modelů představuje mezikrok.

5.2 Přímá vizualizace

Přímá vizualizace volumetrických dat byla ve studiích a článcích daleko méně zastoupena. Tato vizualizační metoda vykresluje přímo surová data ve formátu DICOM, což eliminuje mezikrok tvorby modelů. Na rozdíl od klasického vykreslování modelů se zde používá RayMarching [28] nad volumetrickými daty, která reálně pocházejí z CT/MRI.

Přesnost přímého zobrazení volumetrických dat je jedním z hlavních přínosů tohoto řešení. Vzhledem k tomu, že data nejsou optimalizována, redukována či transformována, je toto řešení kvalitativně srovnatelné s prohlížením CT snímků na monitoru, avšak s přidanou třetí dimenzí.

Je důležité poznamenat, že kvalita zobrazení závisí také na detailnosti CT skenování pacienta, při kterém lze určit, jak podrobně chceme pacienta oskenovat. Je třeba zmínit, že čím detailnější celkový sken je, tím více je pacient obvykle ozářen. Jelikož jsou data z CT skenů nespojitá a snímky jsou pořizovány v předem nastavených intervalech, musí při přímém zobrazení probíhat interpolace dat, což je znázorněno na obrázku 5.1. Nejběžnějším typem interpolace je trilineární. Kvalitnější, avšak náročnější interpolace, je trikubická.



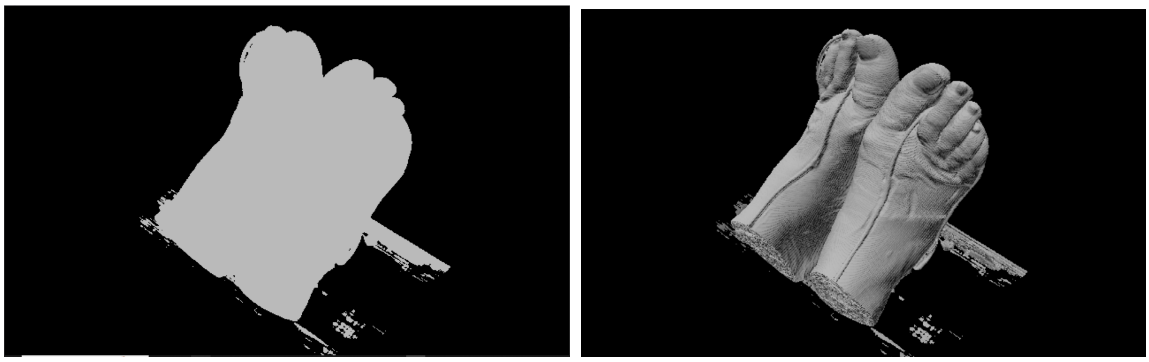
Obrázek 5.1: Znázornění interpolace mezi CT snímky (převzato z [29])

Nespornou výhodnou přímého zobrazení je pak možnost prohlédnout si data i bez provedené segmentace orgánů. V přímém zobrazení se primárně pracuje s hustotou, což umožňuje v reálném čase vybírat interval hustoty, ve kterém se data zobrazují. Vzhledem k tomu, že různé orgány mají obvykle různou hustotu [30], lze takto manuálně vybrat orgány které chceme zobrazit. Toto řešení není dokonalé, protože i když je interval hustoty vhodně zvolen, orgány nemají všude stejnou hustotu a dochází k odchylkám. Ve skutečnosti se nikdy zcela nedá vyhnout různým částicím (splňujícím interval hustoty), které nesouvisí s orgánem, který chceme vizualizovat. Proto je i zde segmentace, která ohraničuje oblasti jednotlivých orgánů, velmi žádoucí. V kombinací se segmentací lze pak upravovat interval hustoty přímo v daném segmentu nebo segmentech.

Přímá zobrazovací metoda nabízí celkem tři známé typy zobrazení. Prvním typem je Maximum intensity projection [31] (MIP). Tento druh zobrazení funguje tak, že si paprsek během své cesty daty pamatuje maximální hodnotu, na kterou narazil. V případě CT skenu jde o Hounsfieldovy jednotky vyjadřující radiologickou hustotu. Viditelné jsou pak části s nejvyšší hustotou. Tato technika se hodí k vizualizaci kostí, nebo cizích pevných těles, které se v těle mohly ocitnout, například v důsledku střelného poranění.

Druhou známou zobrazovací metodou je zobrazení povrchu. Na rozdíl od MIP zobrazení, kde se bere maximální hustota v cestě paprsku, používá zobrazení povrchu minimální hustotu. To znamená, že se primárně použijí data, kde paprsek při cestě poprvé narazí na hustotu větší než hustota

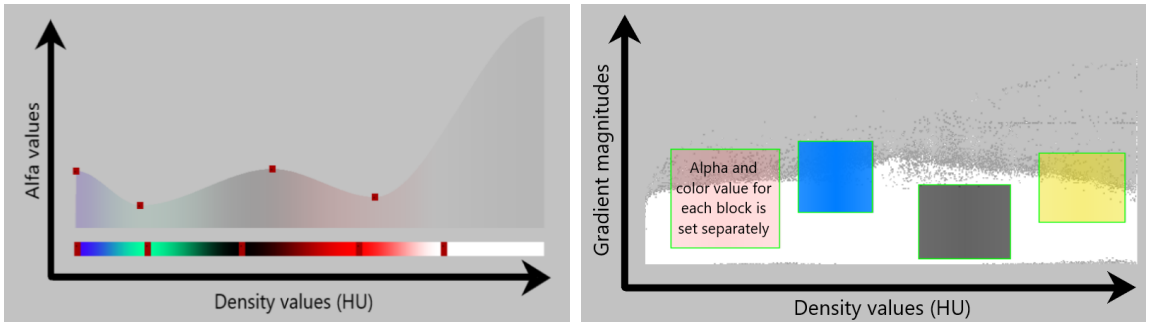
vzduchu. Při tomto typu zobrazení je nutné pro rozumný výsledek použít osvětlovací vzorce s normálami, které lze získat z vypočtu změny gradientu hustoty v daném místě. Zároveň je nutné ve scéně umístit bod, který bude představovat pozici zdroje světla. Díky použití osvětlovací rovnice je pak detailně vidět povrch jednotlivých orgánu. Bez použití osvětlení by nebyly patrné žádné strukturální změny. Toto je znázorněno na obrázku 5.2, kde je pro osvětlení použit Lambertův difuzní model.



(a) Bez osvětlení (b) S použitím lambertiánu

Obrázek 5.2: Srovnání povrchu datasetu s použitým osvětlením

Třetím a nejspíše nejpoužívanějším typem je metoda Direct volume rendering (DVR) [32], která dokáže kombinovat hustotu a průhlednost více orgánu na paprskové trajektorii. Metoda využívá tzv. transfer funkci (TF) pro určení barvy a průhlednosti jednotlivých struktur v daných bodech. Tato funkce musí byt před vykreslováním nastavena. Funkce mohou být různé a mohou se i dimenzionálně lišit podle potřeby. Vizualizační knihovna, o které bude psáno později, například podporuje dva typy transfer funkcí, které jsou vidět na obrázcích 5.3.



(a) 1D transfer funkce (b) 2D transfer funkce

Obrázek 5.3: Ukázka typu transfer funkcí užitých v aplikaci

Je dobré zmínit, že metoda vykreslování povrchu může také využívat transfer funkci k získání barvy. To znamená, že jednotlivé povrchy orgánů budou vhodně barevně odlišeny. Na obrázku 5.4

jsou pak znázorněny všechny tři typy přímého zobrazení

(a) Zobrazení pomocí metody MIP (b) Zobrazení povrchu (c) Zobrazení pomocí metody DVR

Obrázek 5.4: Srovnání metod přímého zobrazení

Vzhledem k tomu, že přímé vykreslování probíhá s voxelovými daty, není v aplikaci problém udělat tzv. okluzní shader, který dokáže voxely vhodně zakrýt podle zvolené vzdálenosti. Tím je možné dosáhnout, že bude možné v datasetu provádět řezy v reálném čase bez nutnosti data jakkoliv modifikovat. U modelů toto představuje problém, neboť by u nich musela proběhnout nová triangulace v místě řezu s případným dopočítáním nových trojúhelníků. Ve srovnání s tím je použití okluzního shaderu k provádění řezů velmi plynulé v reálném čase. Metoda přímého zobrazení má také výhodu, že zobrazení nového datasetu je záležitost jen překopírování nových dat. Není zde žádný mezikrok, jaký je potřeba u modelového řešení. Přímá vizualizace tedy nabízí daleko větší uživatelský komfort pro lidi, kteří budou v budoucnu se softwarem pracovat.

Jako zásadní nedostatek přímé zobrazovací metody jsou nároky na výpočetní výkon, které ve srovnání s modelovým řešením, jsou daleko vyšší. Hlavní zátěž spočívá v procesu RayMarchingu. Naivní přístup posílá pro všechny pixely na obrazovce paprsek, který prochází prostorem. Pokud paprsek narazí na volumetrická data, dojde k jejich vykreslení. Současná řešení v enginech často využívají mesh síť s předem danou velikostí, která má speciální RayMarching shader. Tímto způsobem se eliminuje náročný RayMarching v oblastech, kde data nejsou přítomna. Avšak RayMarching je oproti klasické rasterizaci natolik náročný, že aplikace by pravděpodobně nebyla prakticky použitelná bez externího výpočetního výkonu. Dalším nevýhodou je nutnost implementace RayMarching shaderu nebo dalších shaderů, jako je occlusion shader. Na rozdíl od modelového řešení, kde lze modely jednoduše vložit do scény a automaticky je vykreslit s použitím výchozího shaderu, je zde nutné vytvořit zcela nové shadery. S tím je spojena velká časová náročnost tohoto řešení.

V kontextu použití RayMarching shaderu je důležité zvážit, jak efektivně předat hustotní dataset do shaderu pro realizaci RayMarchingu. Celkově se zde nabízí řešení ve vytvoření 3D Textury, která bude reprezentovat celý dataset. Hodnoty v této textuře pak budou korespondovat s hustotními hodnotami v DICOM datasetu. Tyto hustotní hodnoty je však nutno z DICOM datasetu nějak přečíst. K tomu je vhodné použít již dostupných knihoven pro práci s DICOM formátem, jako je například OpenDicom nebo SimpleITK. Tyto knihovny nabízí poměrně jednoduchý postup extrakce potřebných dat ze snímků, takže díky tomu nepředstavuje vytvoření 3D Textur pro shader zásadní

komplikaci. Nicméně DICOM datasety mohou být velmi detailní s rozsáhlým počtem snímků, což může způsobit, že proces procházení snímků a tvorby 3D textur může trvat značnou dobu. V kontextu běžně dostupného PC hardwaru se jedná o rozmezí desítek sekund až po několik minut.

Pro shrnutí, výhodami přímé vizualizace jsou:

- vysoká přesnost zobrazení s minimální ztrátou detailů,
- segmentace není nutná, avšak je žádoucí,
- možnosti různých typů zobrazení a provádění řezů v datech,
- žádný mezikrok podobný u modelů, a jednoduché zobrazení nových dat.

Do nevýhod pak patří:

- velká výpočetní náročnost,
- nutnost implementace RayMarching algoritmu a specifických shaderů,
- posílání hustotních dat do shaderu.

Kapitola 6

Základní teorie zarovnání modelu

Zarovnání holografického modelu s reálným pacientem a přesnost tohoto zarovnání představuje obecně velký problém. Zarovnání lze rozdělit do dvou kategorií - manuální zarovnání, které provádí sám chirurg, a automatické zarovnání, které se opírá o různé metody.

6.1 Manuální zarovnání

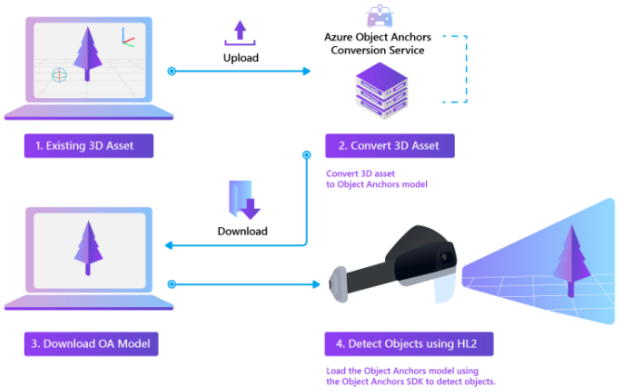
Manuální zarovnání je základní a nejjednodušší způsob zarovnání, který není zpravidla automatizovaný. V praxi to znamená, že uživateli je načten model, který se následně snaží přesně zarovnat s reálným pacientem pomocí specifických přístupných bodů. V tomto ohledu má přímá zobrazovací metoda značnou výhodu, neboť kromě konkrétních orgánů je možné zobrazit i povrch kůže daného pacienta, který lze viditelně zarovnat s reálnou kůží a specifickými body. Po zarovnání lze kůži vypnout a zobrazit již pouze orgány. V případě nepřímé metody by modely pravděpodobně musely obsahovat v určitých místech uměle vytvořené body vycházející ze specifických míst v datasetu, které jsou zároveň viditelné navenek. Model by se poté dal s pacientem zarovnat pomocí těchto uměle vytvořených bodů. Hlavní nevýhodou manuálního řešení je však jeho pracnost. Model se musí po každém spuštění aplikace znovu zarovnat, což zabírá čas. Přesnost takového zarovnání závisí na tom, jak pečlivě se uživatel procesu bude věnovat, což znamená kompromis mezi časem potřebným pro zarovnání a přesností zarovnání.

6.2 Automatické zarovnání

Automatické zarovnání přináší řadu výhod a je to způsob zarovnání, kterého bychom rádi dosáhli. Bohužel dosud nebyl nalezen ideální způsob jeho provedení. V současnosti existuje několik přístupů k řešení tohoto problému.

6.2.1 3D detekce objektu

3D detekce objektu spočívá v tom, že zařízení předem dostane referenční model, který má v prostředí hledat. V případě nalezení objektu se ho pokusí překrýt hologramem, přičemž se berou v úvahu pozice, rotace a měřítko. Zařízení Hololens 2 nabízí tuto možnost pod názvem Azure Object Anchor [33]. Microsoft dokumentace poskytuje kvalitní tutoriál [34] pro tuto službu. Avšak využití Object Anchors má zásadní nevýhodu, kterou je proces nastavení pro každý nový model, který chceme rozpoznat. Nejprve se musí vytvořit tradiční model, který se následně převede do formátu kompatibilního s HoloLens 2. Po převedení a nahrání modelu na Hololens 2 by mělo být zařízení schopno rozpoznat daný model v prostoru. Celý postup je znázorněn na obrázku 6.1. Jak si lze představit, tento dodatečný postup značně snižuje myšlenku jednoduchosti zobrazení pomocí přímé metody, která nevyžaduje vytváření modelů.



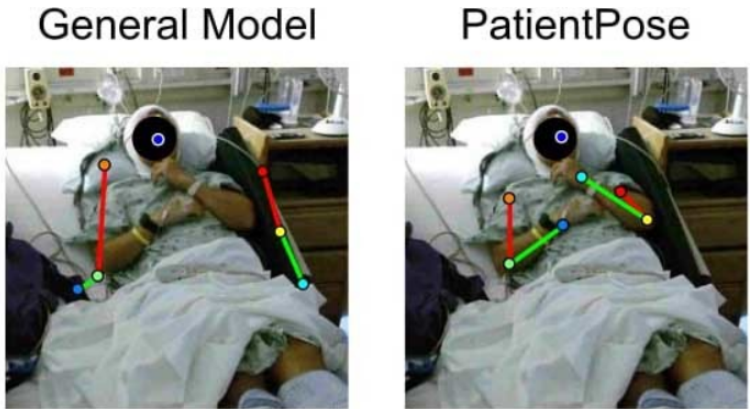
Obrázek 6.1: Princip detekce pomocí Azure Object Anchors (převzato z [33])

Také přesnost vytvářené prostorové mapy, která se používá k detekci objektů, se může u zařízení lišit. Další nevýhodou je pak celková náročnost na výkon tohoto řešení, kde musí zařízení kontinuálně skenovat prostor pro aktualizace prostorové mapy a hledat v ní objekty. O Object Anchors bude více řečeno v implementační kapitole v sekci 7.3.1.

6.2.2 Automatická detekce pozice pacienta

Automatická detekce pozice pacienta se liší od detekce modelů tím, že pracuje přímo s pacientem. Většina současných implementací této detekce (např. HRNet, OpenPose, AlphaPose) vychází z použití neuronových sítí detekující jednotlivé části lidského těla. Tento obecný postup odstraňuje zásadní nedostatek předchozí metody, kde bylo pro každou novou detekci nutné vytvořit nový model, zpracovat ho, atd. Avšak nevýhodou je zde extrémní výpočetní náročnost pro aktualizace pozice pacienta. Nebyla nalezena žádná studie nebo článek využívající tuto detekci pacienta na operačním sále pro zarovnání modelu. Nejblíže byl článek [35] zkoumající možnosti detekce pacientovy pozice

v klinických prostředích. Použití je znázorněno na obrázku 6.2. Tento způsob zarovnání pravděpodobně ještě nebyl proveden v praxi a vychází čistě z teoretických předpokladů proveditelnosti.



Obrázek 6.2: Automatická detekce pozice pacienta (převzato z [35])

6.2.3 2D detekce markerů

2D detekce obrázků a jiných speciálních markerů, jako jsou QR kódy nebo čárové kódy, se vyznačují vysokou popularitou. Výhody této metody spočívají v relativně vysoké přesnosti a nízké výpočetní náročnosti. Po úspěšné detekci odpovídajícího QR kódu lze aktualizace pozastavit, aby se předešlo zbytečné detekci nových markerů. Nicméně, tato metoda má svá omezení. Pro úspěšné fungování je nezbytné, aby pacient ležel v předem stanovené pozici a jsou povoleny jen minimální odchylky. Důvodem je, že holografický model není navázán na pacienta, ale na nehybný QR kód, který je fyzicky upevněn v prostředí.

6.2.4 Prostorové umístění

Metoda prostorového umístění spočívá s využitím prostorové mapy, podobně jako při detekci modelů, avšak s tím rozdílem, že dochází k detekci celé mapy jako takové. U zařízení jako je Hololens 2, se o tvorbu prostorové mapy stará několik kamer na zařízení. Díky schopnosti zařízení vytvořit takovou mapu prostředí si ji může zařízení uložit do paměti. Při novém načtení aplikace v zapamatované místnosti pak dochází k rozpoznání prostředí a lze využít této prostorové mapy k ukotvení modelu, který bude umístěn do předem určené pozice v místnosti. U zařízení Hololens 2 se taková služba jmenuje Azure Spatial Anchors [36].

Toto řešení má však své nevýhody. Pokud zařízení používáme v podobně konstruovaných místnostech, může se zařízení splést, což může vést k tomu, že momentálně detekovaná místnost bude považována za jinou. S tímto se také pojí různé artefakty při ovládání, kde se detekce rukou může samovolně resetovat a holografické objekty v místnosti různě uskakovat. Další nevýhodou tohoto řešení je, že se místnost nesmí nijak měnit. Problém nastává, pokud jsou v místnosti židle a další

přemístitelné objekty, které po přemístění mohou zařízení zmást natolik, že v lepším případě bude docházet k již zmíněné nestabilitě holo-objektů. V horším případě se místnost nepodaří vůbec detekovat.

6.2.5 Zarovnání pomocí dodatečného CT skenu

U této metody je pacient znehybněn a umístěn do CT tunelu, kde je proveden dodatečný sken pro zarovnání. Tento postup je znázorněn na obrázku 2.6. Řešení bylo také zmíněno ve studii [11] a úspěšně provedeno v článku [6]. Hlavní výhodou této metody je vysoká přesnost zarovnání, která se podle článku [6] pohybovala v řádu jednotek milimetrů. Autoři využili tuto přesnost k minimalizaci velikosti prováděného řezu.

Avšak zásadní nevýhodou tohoto řešení je dodatečné vystavení pacienta radiaci při použití nového CT skenu. Po provedeném skenu se také s pacientem v průběhu operace nesmí hýbat. Další nevýhodou jsou také dodatečné náklady na tento sken. Alternativní možností zarovnání je použití MRI, které nezpůsobuje ozáření pacienta.

Kapitola 7

Implementační část

V této kapitole je popsána především implementační část práce, tvorba aplikace a celková realizace řešení. Implementaci zásadně zkomplikovalo pozdní dodání finálních materiálů pro vývoj ze strany FNO, které byly dodány až zhruba v polovině ledna 2023, což značně omezilo vývojový čas.

7.1 Základ

Jak již bylo zmíněno, aplikace je vyvíjena v prostředí Unity. Vzhledem k mým celkovým zkušenostem s Unity a jazykem C#, byla tato volba poměrně jasná. Unity se spolu s Unreal Enginem stále řadí na špičku v oblasti herních enginů. Oba tyto enginy mají své silné a slabé stránky. Dalo by se říct, že Unreal dominuje v projektech s cílem na fotorealistickou grafiku. Na druhé straně, Unity dominuje v obrovské podpoře ze strany dostupných tutoriálů a rozsáhlého Asset Storu, který obsahuje kvalitně zpracované nástroje všeho druhu. Oba enginy jsou také často aktualizovány. Unity se konkrétně řídí politikou hlavních třech velkých verzí ročně a menšími verzemi zaměřenými především na opravu problémů. Jejich dlouhodobé (Long-term support - LTS) verze znamenají minimálně 2 roky plné podpory s opravami chyb a přidáváním vylepšení.

Významný milník pro Unity by v tomto roce mělo znamenat vydání plné verze Unity DOTS entity systému, který je v současné době v testovací verzi. DOTS umožňuje využívat datově orientovaný přístup, který přináší velké výkonnostní výhody v částech aplikace, kde je potřeba starat se o velkou skupinu nezávislých objektů stejného typu. Tento přístup může být užitečný například pro různé simulace velkého počtu fyzikálních objektů, simulací armád a podobně. Datově orientovaný přístup se dá také kombinovat s klasickým přístupem, kde DOTS entity jsou kompatibilní s klasickými GameObjecty v Unity.

Výběr AR knihovny pro zařízení Hololens 2 byl poměrně jasný. Na trhu existuje několik knihoven ([37], [38], [39]), které umožňují práci s tímto zařízením, avšak tyto knihovny se často liší svými funkcemi a vlastnostmi. V této práci bylo jednoznačně vhodné využít knihovnu Mixed Reality Toolkit (MRTK) [40] od Microsoftu, která byla vyvíjena přímo pro účely použití na Hololens 2.

Knihovna MRTK nabízí celkem velké množství předpřipravených ovládacích prvků a podporu pro vhodnou interakční logiku s objekty ve scéně. MRTK ve srovnání s ostatními knihovnami obsahuje těchto prvků pravděpodobně nejvíce a je vhodné jí využít jako základ, na kterém se postupně bude dál stavět. Knihovna obsahuje i další specifické služby, jako jsou Azure Object Anchors, Azure Spatial Anchors a další, které využívají Azure cloudových služeb. Některé z těchto služeb jsou zatím stále v experimentálním módu, ale dají se již převážně využít pro vývoj.

Výhodou je, že pro účely této práce jsou všechny potřebné služby z MRTK knihovny zdarma, případně u cloudových služeb se zanedbatelnou cenou v tarifu „pay as you go“, což je oproti konkurenci rozdíl, která často za podobné služby požaduje velmi vysoké sumy. Příkladem je třeba knihovna Vuforia [37], kde se placená licence může pohybovat od několika set dolarů až po několik tisíc za rok. I když se někdy může zdát, že MRTK knihovna danou službu zatím nepodporuje, je funkcionalita často jen skrytá, nebo v experimentálním módu a lze ji dodatečně aktivovat. Některé specifické funkční prvky, jako je například detekce QR kódu zatím nejsou součástí hlavního balíčku MRTK a jsou dostupné samostatně. Použití MRTK knihovny v těchto specifických oblastech má stále různé nedostatky a často je nutné počítat s komplikacemi. Nicméně je to pochopitelné, vzhledem k tomu, co všechno knihovna nabízí. Je třeba také brát v potaz, že se jedná o relativně novou knihovnu, na které se stále aktivně pracuje a vyvíjí se v ní mnoho věcí současně. Základní funkcionality, které MRTK knihovna poskytuje, jsou však zpracovány kvalitně a většinou se dají použít bez problémů.

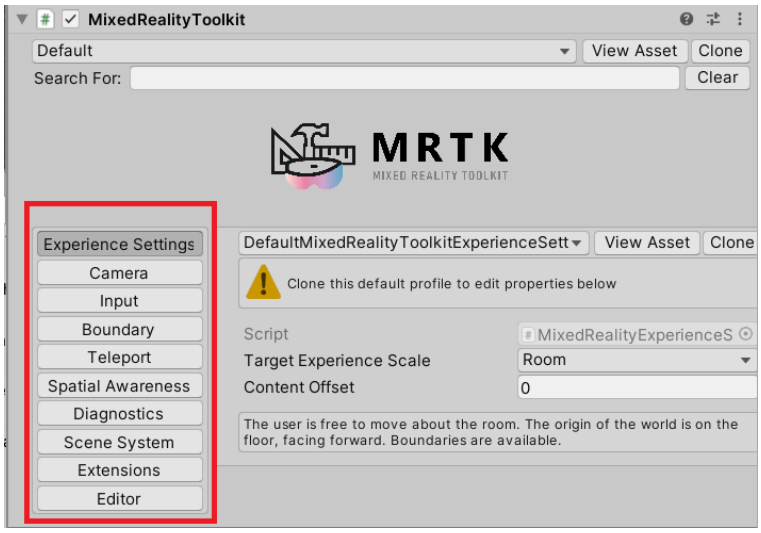
Pro práci s knihovou je k dispozici kvalitně zpracovaná sekce tutoriálů. Tutoriály obsahují vhodné obrázky a jednotlivé kapitoly uživatele postupně provádějí od základů až k složitějším úlohám. Autoři vynaložili velké úsilí pro správnou integraci knihovny MRTK do Unity enginu, a mnoho MRTK prvků používá užitečné Unity komponenty jako jsou skriptovací objekty, Unity události a další. Unity editor je navíc vhodně využit, což umožňuje snadnou konfiguraci a úpravu prvků knihovny.

Společně se založením Unity projektu v rámci této práce byl pro něj vytvořen také veřejný repozitář na GitHubu, kde se celý projekt nachází. Odkaz na repozitář je také uveden v příloze práce. V pozdějších sekcích bude na tento repozitář odkazováno.

7.1.1 Nastavení projektu pro AR a Hololens 2

Nastavení projektu bylo realizováno s využitím nástroje Mixed Reality Feature Tool [41] a dostupného návodu [42]. Feature Tool umožnuje správnou instalaci knihovny do již existujícího projektu. V tomto nástroji si lze vybrat, které komponenty budou v knihovně použity, aby se zabránilo instalaci nepotřebných prvků. Nástroj společně s instalací komponent nainstaluje také všechny potřebné závislosti.

Hlavní nastavení projektu pak probíhá v objektu „Mixed Reality Toolkit“, který je nutno vložit do scény. Tento globální objekt je zobrazen na obrázku 7.1, kde lze také vidět nastavitelné komponenty.



Obrázek 7.1: MRTK objekt ve scéně a jednotlivé komponenty

V rámci daných komponent se nastavuje typ relace, plugin poskytovatelé jednotlivých komponent, pozadí aplikace, slova pro hlasové ovládání, diagnostické systémy a další prvky. V aplikaci je pro většinu funkcí použit OpenXR poskytovatel, který umožňuje lepší multiplatformní podporu. Tento systém nastavení využívá uživatelské profily realizované pomocí skriptovacích objektů. Profily jsou ve výchozím stavu již nastavené a pokud je chceme jakýmkoliv způsobem měnit, je nutné tyto výchozí objekty naklonovat. Naklonované objekty již lze upravovat. Při běhu aplikace je pak možné prostřednictvím skriptu celkový profil vyměnit. Pokud tedy chceme v hlavním nastavení provést jakékoliv změny v průběhu běhu aplikace, je nutno cílové profily předem připravit.

7.1.1.1 Nastavení vzdáleného připojení zařízení

Pro efektivní práci je vhodné v projektu nastavit vzdálené připojení zařízení, které se u Hololens 2 nazývá Holographic Remoting [43] (dále v textu zkráceně jako remoting). Tím se ušetří čas, který by jinak byl stráven sestavováním aplikace po každé změně. Pomoci remotingu se Hololens 2 zařízení, na kterém běží remoting aplikace, dokáže pomocí IP adresy spojit s počítačem a změny v kódu zobrazit rovnou v Unity herním módu. Při běhu aplikace se do herního módu přenáší vstupy z Hololens 2, jako je detekce rukou, pozice zařízení, pozice očí atd., a do Hololens 2 zařízení se pak přenáší obraz z herního módu. Pro správné nastavení remotingu existuje v dokumentaci dedikovaná sekce [44], kde je popsán nutný postup k jeho zprovoznění.

U VR vývoje, konkrétně u zařízení Quest 2, se podobný krok řeší pomocí Quest Linku, nebo Quest Air Linku, což umožňuje úplné propojení VR zařízení s počítačem. Po tomto propojení je zařízení schopno automaticky detekovat tzv. XR inicializaci, kde v případě spuštění Unity herního módu se mód společně se zařízením automaticky přepne do VR režimu.

Pro ladění aplikace je vzdálené připojení klíčové, protože jsou tak vývojářům k dispozici všechny důležité ladící prvky Unity enginu, jako například výkonnostní profiler, snímkový debugger, vstupní debugger, fyzikální debugger a další. Důležitou výhodou je také možnost využití klasického krokování v kódu. Tyto aktivity je výrazně těžší vykonávat v konkrétním sestavení aplikace, kde kvůli Universal Windows Platform (UWP) by ladění muselo proběhnout výhradně přes Visual Studio, které není pro některé specifické prvky plně připraveno.

Projekt využívá Unity verzi 2020.3x dle doporučení z MRTK stránky. Nově byla stránka aktualizovaná a je již možné plně využít verzi 2021.3x. Původně stránka doporučovala starší verzi z důvodu určitých chyb přítomných v pozdějších verzích.

7.2 Dosažení stavu semestrálního projektu

Tato práce vznikla zejména s cílem vyzkoušet možnosti medicínské vizualizace v dnešních moderních enginech. Před zahájením této práce existoval semestrální projekt, který se zabýval možnostmi medicínské vizualizace. Autorem semestrálního projektu je Ondřej Fojtík, který velkou část nástrojů vytvořil sám a nepoužíval téměř žádné již existující knihovny. Na začátku této práce bylo v prvé řadě nutné dostat projekt do stavu, který by odpovídal dokončené semestrální práci. Cílem tedy bylo vyvinou aplikaci se stejnými funkcemi a vlastnosti jako měl semestrální projekt, avšak s použitím Unity.

Základní premisou semestrální práce bylo vyvinutí aplikace, schopné vhodně zobrazit model jater na základě CT skenu ve fyzické kolébce s pomocí QR markeru. Tento nápad vznikl během vývoje semestrálního projektu při dialogu s chirurgickou klinikou FNO. V modelu bylo možné jednotlivé CT snímky posouvat, přičemž zobrazení snímku způsobilo zneviditelnění zbývající části modelu, aby byl snímek lépe viditelný. Snímky sloužily pouze jako dočasný prvek pro testování a nebyly založeny na konkrétního modelu.

Zobrazení jednotlivých snímků v této práci nebyl obecně problém, ale bylo nutné najít způsob, jak zakrýt zbývající část modelu v závislosti na pozici zobrazovaného snímku. Existují dva klasické způsoby, jak toho dosáhnout. První způsob je napsání vlastního shaderu, do kterého by se předala pozice zobrazovaného snímku a konkrétní vrcholy modelu by nebyly vykresleny, pokud by se nacházely za daným snímkem. Problémem však je, že celá mesh síť by takto obsahovala díry v místě za snímkem. Druhé řešení, a které již bylo použito, spočívalo v úpravě mesh sítě. Pro tento účel byla využita užitečná open-source knihovna OpenFracture s licencí MIT, která umožňuje v Unity pracovat s mesh sítí při běhu aplikace. Knihovna podporuje roztříštění (fracturing) ale i kontrolovatelné rovinné řezy v modelu a umožňuje vyplnit prostor v místě řezu novými vrcholy k zaplnění díry vybraným materiálem.

Při spuštění aplikace se výchozí mesh síť tedy nařezala na základě pozic jednotlivých snímků, a při běhu aplikace již docházelo pouze k vypínání a zapínání vhodných nařezaných částí společně s CT snímkem pomocí přidaného posuvníku. Výhodou tohoto řešení bylo, že náročné výpočty se provedly

již na začátku, a při běhu aplikace probíhalo již standardní vykreslování bez dalších výpočtů. K tomuto bylo třeba knihovnu lehce modifikovat, neboť v původním stavu nebyla uzpůsobena pro hromadné řezání celkové mesh sítě. Docházelo k vytváření duplicitních částí, což nebylo žádoucí. Po úpravě metod s přidanou možností zachovat pouze jednu ze dvou částí řezu se aplikace dostala do podoby srovnatelné s výstupem ze semestrální práce. Tuto podobu pak lze vidět v tomto videu. Umístění modelu do kolébky pak probíhá pomocí QR kódu umístěného na kolébce. Pro aplikaci proběhly patřičné ukázky a byla otestována několika lidmi. Další postup práce poté mohl následovat.

7.3 Zarovnání modelu ve scéně

V této práci byly primárně vyzkoušeny dva způsoby zarovnání: pomocí QR markerů a pomocí Azure Object Anchors. Ve finální verzi aplikace jsou použity pouze QR kódy.

7.3.1 Testování Object Anchors

Testování Object Anchors proběhlo během během vývoje popisovaném v sekci 7.2. Testy vycházejí z návodu, dostupného v této sekci [34] dokumentace a z ukázkového repozitáře, ze kterého se dá vyjít a snadno si věci vyzkoušet. Object Anchors jsou spojeny s cloudovou službou Azure, a proto je nutné mít pro jejich testování Azure účet. Azure nabízí dvouměsíční bezplatné zkušební období, během kterého si lze vyzkoušet tyto i další cloudové služby. Po dvou měsících by v tomto případě byly náklady v tarifu „pay as you go“ minimální, neboť by služba byla použita ve velmi malém měřítku. Cloudové služby jsou zde využity zejména pro převod klasických modelů do formátu OA, který umožňuje zařízení Hololens 2 rozpoznávat objekty v prostoru.

Pro celkový převod je nutné použit ukázkový Visual Studio projekt, který je k dispozici z již zmíněného repozitáře ve složce conversion. Použití projektu vychází ze správného nastavení třídy Configuration.cs, která se v projektu nachází. Tato třída obsahuje všechny potřebné informace pro připojení k Azure a cesty k lokálním souborům. Každý nový převod vyžaduje, aby uživatel tento projekt ve Visual Studiu otevřel a upravil tuto třídu. Obecně by nebyl problém projekt upravit a převést ho například do Windows Forms nebo Windows Presentation Foundation (WPF), což by umožnilo uživatelsky přívětivější ovládání i pro neprogramátory. Proces převodu a nahrání na Cloud tedy není ve výchozím stavu úplně ideální. Celkově popisovaný postup převodu je také uveden v dokumentaci [45].

Během testování bylo zjištěno, že přesnost rozpoznání není zcela dostačující. Služba pravděpodobně funguje skvěle na větší objekty, avšak u menších objektů se potýká s problémy způsobenými nedostatečnou přesností interně vytvořené prostorové mapy. V dokumentaci je uvedeno, že služba je určena pro objekty s minimálními rozměry jednoho metru v každé ose [33]. Informace je zásadní, protože pokusy s kolébkou vytištěnou na 3D tiskárně, která měla zhruba poloviční rozměry selhaly a objekt v podstatě nebyl rozpoznán. Byly testovány i další objekty, avšak pro ně nebyly

k dispozici přesné 3D modely. Modely pro detekci byly vybrány co nejpodobnější, ale pro ideální fungování služby musí mít modely přesné rozměry odpovídající reálnému objektu. Přesto se detekce objektů s přibližnými modely dařila, což lze vidět na ukázkovém videu. Na videu je nicméně patrný již zmíněný problém s neodpovídajícími rozměry.

7.3.2 Implementace zarovnání dle QR kódu

Ve finální verzi aplikace je zarovnání modelu ve scéně realizováno s využitím QR markerů na kolébce vytisknuté pomocí 3D tiskárny. Tuto funkcionalitu MRTK knihovna ve výchozím stavu neobsahuje. Funkcionalita je dostupná z QRCode repozitáře. Na hlavní stránce repozitáře jsou uvedeny důležité prerekvizity a návod k zprovoznění. Po zprovoznění je aplikace schopna rozpoznat QR kód a na jeho pozici umístit zvolený prefab[1]. Pozice jaterního modelu byla v prefabu manuálně nastavena tak, aby co nejvíce odpovídala reálnému uložení ve fyzické kolébce. Po detekci QR kódu při běhu aplikace tak dojde k vhodnému umístění modelu do fyzické kolébky. Díky otevřenosti MRTK knihovny a tohoto repozitáře bylo možné věci lehce upravit dle vlastních potřeb.

Během testování bylo zjištěno, že díky kontinuální lokalizaci QR kódu může jeho pozice v aplikaci v důsledku těchto aktualizací lehce poskakovat. V případě, že QR kód má být v průběhu běhu aplikace nehybný (což platí pro kolébku), lze tomuto problému zabránit vypnutím QR aktualizací po prvotní detekci. Uživatel si v aplikaci může zvolit, zda chce či nechce aktualizace QR kódů. V této práci je vypínání řešeno pomocí rozhraní v části kódu 7.1, které implementují čtyři třídy z QR trackeru a obsahují vypnutí aktualizací pro konkrétní třídy.

```
public interface IQRUpdate
{
    public void EnableQRUpdate(bool value);
}
```

Listing 7.1: QR rozhraní

Díky tomu, je pak možné na všechny potřebné objekty zavolat globálně vypnutí/zapnutí aktualizací QR kódu, což je vidět na kódu 7.2.

```
public void EnableQRUpdates(bool value)
{
    foreach (IQRUpdate i in FindObjectsOfType<MonoBehaviour>().OfType<IQRUpdate>())
        i.EnableQRUpdate(value);
}
```

Listing 7.2: QR Update metoda

Kód pro detekci QR kódů má jednu specifickou vlastnost. Autoři repozitáře se rozhodli, že se prefab automaticky bude pokládat i do starých oblastí detekcí QR kódu. Vzhledem k tomu, že

[1] Prefab v Unity je znovupoužitelný asset, složený z jednoho nebo více herních objektů (GameObjects) a jejich komponent. Hlavní výhodou je snadné vytvoření a možnosti jej využít napříč scénami. Prefab je uložen v samotném projektu, což umožňuje snadnou manipulaci a znovupoužití.

zařízení Hololens 2 ukládá QR kódy na systémové úrovni, nedochází ke smazání QR záznamů při vypnutí aplikace, ale až při restartu zařízení. Vysvětlení tohoto chování je uvedeno v této sekci [46] dokumentace a primárním důvodem je zabránění vzájemnému rušení jednotlivých aplikací také využívajících QR kódy. Toto chování však způsobuje, že při každém novém spuštění aplikace, pokud nedošlo k restartu zařízení, se prefab automaticky vloží na staré pozice, které mohou být neaktuální. Tento problém lze vyřešit kontrolou času detekce QR kódu, kde každý QR kód má u sebe uvedenou poslední zaregistrovanou detekci. Pokud tedy zabráníme vložení prefabu pro starší detekce, než je datum a čas spuštění aplikace, je tento problém vyřešen. Řešení je obecně uvedeno v komentáři, který se nachází v otevřeném problému repozitáře a kde uživatelé řešili stejný problém.

QR repozitář má ještě jeden zásadní problém, spočívající v specifických chybách, pokud došlo k sestavení aplikace pro platformu Windows standalone. Vzhledem k tomu, že aplikace je nakonec cílena právě na tuto platformu, bylo vyřešení tohoto problému naprosto zásadní. Později, po objevení tohoto uzavřeného vlákna, které se podobalo zdejšímu problému, bylo nakonec zjištěno, že hlavním viníkem je pár chybějících DLL souborů v kořenovém adresáři. V již zmíněném repozitáři jsem otevřel nové vlákno, které obsahuje dočasné řešení odkazující se hlavně na již uzavřený problém. Domnívám se však, že tento problém by neměl být uzavřen, dokud nebude náležitě opraven, aby ostatní uživatelé, kteří by mohli mít stejný problém, mohli využít dočasné řešení problému.

Prvním zásadním problémem je chybějící DLL soubor spojený s knihovnou WinRT. V repozitáři není zmíněna jeho důležitost, avšak instalace NuGet balíčku obsahujícího tento soubor je pro správnou funkci klíčová. Druhým problémem je špatné nastavení cest, kde určité DLL soubory v konkrétním adresáři je nutno po sestavení překopírovat do kořenového adresáře pro správnou funkčnost. Pravděpodobně není daný adresář, odkud je nutné DLL soubory překopírovat, úplně zahrnut v řešení. V normálním stavu by žádné z těchto zmíněných řešení nemělo být potřeba. Avšak do doby, než bude problém vyřešen, jsou tyto dočasné kroky důležité.

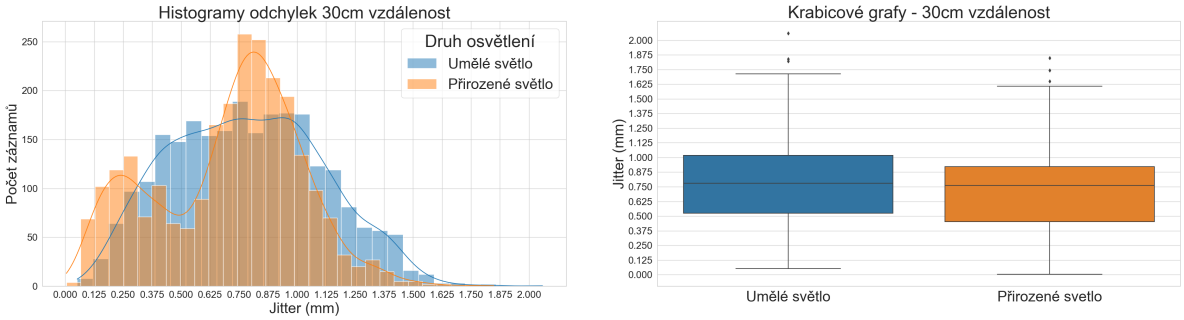
7.4 Přesnost rozpoznání QR kódu a odchylky

Přesnost služby Azure Object Anchors se zde nezkoumá, neboť by pro potřebné měření byl potřeba větší vytisknutý 3D model, jak již bylo zmíněno v podsekci 7.3.1. Měření bylo provedeno pouze pro detekci pomocí QR kódu a zabývalo se odchylkami umístění QR kódu ve scéně (jitter).

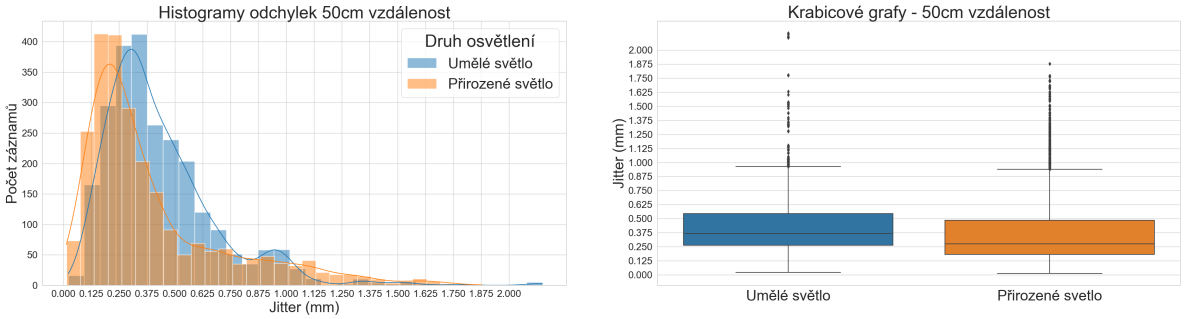
Měření proběhlo ve dvou různých světelných podmínkách: s umělým světlem a s přirozeným světlem. V obou případech nebylo světlo přímé ani oslnivé. Zařízení Hololens 2 bylo umístěno do stabilní pozice naproti QR kódu. QR kód použitý pro měření byl čtvercový s délkou strany přibližně 8 cm. Bylo zjištěno, že zařízení spolehlivě detekuje QR kód o této velikosti ve vzdálenosti přibližně 30 cm až 80 cm. Mimo tento interval již zařízení mělo s detekcí problém. Měření odchylek proběhlo ve vzdálenostech 30 cm, 50 cm a 80 cm od QR kódu a při měření se dbalo na to aby otřesy podlahy byly v době měření minimalizované.

V Unity se zachovalo nastavení, aby jedna jednotka odpovídala jednomu metru v reálném světe. Jelikož byly QR kód a zařízení ve stabilní poloze, lze měřit rozdíly v digitálních polohách, které se mohou i přes fyzickou stabilitu měnit. Odchylky byly zaznamenány dle polohy QR kódu vůči zařízení Hololens 2, pracovalo se tedy v Hololens 2 lokální soustavě. Záznamy byly pořizovány desetkrát za sekundu v průběhu pěti minut pro každé měření. Na grafech pak jednotlivé odchylky znamenají vzdálenosti od výchozí pozice, proto jsou vždy kladné.

Výsledky měření ve vzdálenosti 30 cm od zařízení ke QR kódu lze vidět na sérii grafových obrázků 7.2, výsledky ve vzdálenosti 50 cm lze pak vidět na sérii grafových obrázků 7.3 a výsledky ve vzdálenosti 80 cm na sérii grafových obrázků 7.4.

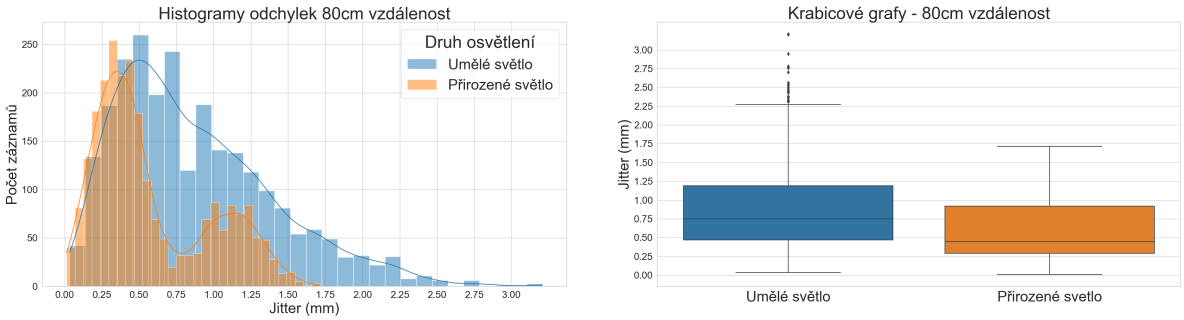


Obrázek 7.2: Vzdálenost 30 cm při kontinuálním rozpoznávání

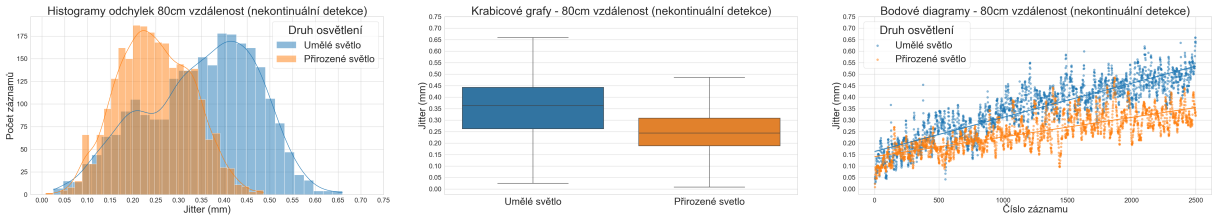


Obrázek 7.3: Vzdálenost 50 cm při kontinuálním rozpoznávání

Na grafových obrázcích 7.5 jsou pak zobrazena měření odchylek bez kontinuálního rozpoznávání QR kódu. Kód byl na začátku běhu aplikace rozpoznán a poté bylo již rozpoznávání vypnuto. U těchto měření je zahrnut také bodový graf, neboť ukazuje zajímavé zjištění, že pokud nedochází ke kontinuální detekci, QR kód má v průběhu času tendenci zvětšovat odchylku a tzv. „driftovat“ pryč. Vidět je to zejména na jednotlivých regresních čarách (lineární regrese). Z bodového grafu jde vidět, že tato tendence byla přítomná u umělého i přirozeného osvětlení, avšak nedocházelo k ní u ostatních měření, proto u nich nejsou bodové grafy uvedeny.



Obrázek 7.4: Vzdálenost 80 cm při kontinuálním rozpoznávání

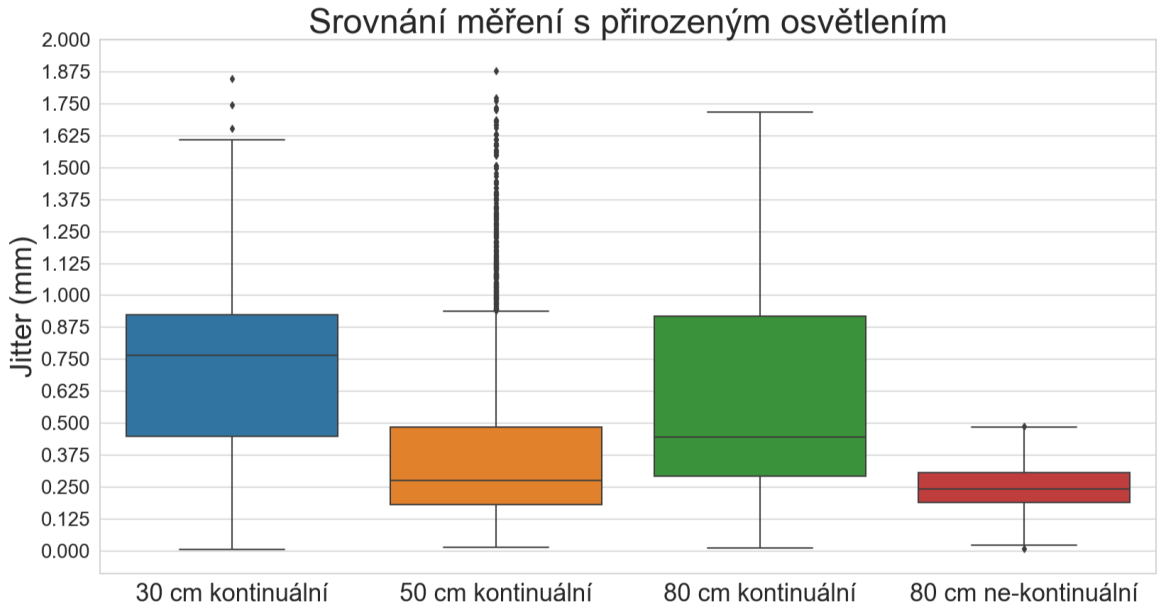


Obrázek 7.5: Vzdálenost 80 cm **bez** kontinuálního rozpoznávání

Z výsledků je patrné, že menší odchylky nastaly s využitím přirozeného světla. To může mít celou řadů důvodu, některé jsou zmíněny přímo v dokumentaci. Z výsledků také vyplývá, že zařízení nejlépe rozpoznávalo polohu QR kódu ve vzdálenosti 50 cm s přirozeným světlem. Měření nicméně ukazuje, že kontinuální rozpoznávání QR kódů vede k větším nepřesnostem. To je vidět zejména ve srovnání měření z obrázku 7.4 s měřením na obrázku 7.5. V případě, že máme nehybný QR kód, je výhodné využít variantu prvotního rozpoznání kódu, kde si zařízení pozici zapamatuje a dále se již nesnaží aktualizovat a rozpoznávat další QR pozice. Na obrázku 7.6 jsou následně srovnány všechny krabicové grafy z měření s přirozeným světlem, kde jde jasně vidět, že díky této metodě se dosahuje lepší přesnosti umístění objektu ve scéně s menšími odchylkami.

Toto tvrzení je také podpořeno následujícími výsledky. Vzhledem k tomu, že všechny datasety měření nemají dle Shapiro-Wilkova testu normální rozložení (p hodnota <0,001), zamítáme podle Leveneho testu (statistika = 521,7, df = 3, P-hodnota <0,001) předpoklad o shodě rozptylů. Na základě Kruskal-Wallisova testu (statistika=2629,7,df=3, P-hodnota <0,001) je zde zamítnuta hypotéza o shodě mediánů. Dle Dunnové post hoc analýzy lze konstatovat, že výrazný rozdíl mediánů byl nalazen u srovnání všech dvojic měření (p hodnoty <0,001). Měření 7.5 však dosahuje nejmenšího mediánu ze všech provedených měření.

Měření se zabývala pouze možností lokalizace objektu ve scéně a přesností takovéto lokalizace pomocí QR kódu. Reálné odchylky způsobené pohybem uživatele a tzv. „drift“ objektů způsobený nedokonalostmi vytvořené prostorové mapy, nedostatkem senzorových dat ze zařízení a dalšími faktory tato měření nezkoumala. O těchto a dalších problémech je psáno v dokumentaci [47].



Obrázek 7.6: Srovnání odchylek s přirozeným osvětlením

7.5 Volumetrické řešení (Metoda přímého zobrazení)

Vývoj volumetrické verze aplikace proběhl na volumetrické větvi, kterou lze nalézt na této adrese. Celkový průběh vývoje je v jednotlivých commitech dokumentován a popsán. Větev také obsahuje návod v souboru **README**, kde se nachází sekce týkající se použití aplikace a sekce pro zprovoznění projektu. Aplikace z této větve využívá kvůli vysoké výpočetní náročnosti externího výpočetního výkonu z připojeného počítače, podobně jak již bylo dříve zmíněno.

7.5.1 Zkoumání RayMarching knihoven pro Unity

V prvé řadě bylo nutné zjistit, která knihovna pro volumetrické vykreslování bude pro aplikaci nejvhodnější. Většinu dostupných knihoven pro Unity lze nalézt na Asset Storu. V kapitole 2 jeden zdroj zmiňoval využití právě této knihovny pro přímé vykreslování. Nicméně knihovna již delší dobu nebyla aktualizována.

Většina knihoven týkající se daného tématu je na Asset Storu zpoplatněna. Z průzkumu knihoven bylo zjištěno, že prakticky jediná aktivní knihovna je knihovna od autora Matiase Lavika. Knihovna je také k dispozici zdarma na GitHubu a je distribuovaná pod licencí MIT. Asset store pak obsahuje poslední kompilovanou verzi této knihovny. Po následném průzkumu se ukázalo, že úroveň knihovny je velmi kvalitní s mnoha možnostmi využití. Navíc, na GitHubu se zdá být tato knihovna jednou z nejpokročilejších. Autor knihovny je na GitHubu aktivní a přístupný k různým dotazům a problémům ze strany uživatelů, komunita je také aktivní. Autor je také otevřen návrhům na vylepšení a dovoluje lidem do knihovny vhodným způsobem přispět.

Tato knihovna je tedy v této práci využita pro možnosti přímého vykreslování. Knihovna RayMarching provádí ve vlastním speciálním shaderu. Ve scéně je pro každý jednotlivý CT dataset vytvořen kvádr s materiálem využívající tento shader (kvádr se dá brát jako bounding box datasetu). Hustotní data jsou převedeny do 3D textury, která je následně poslána do shaderu. Tato 3D textura zachovává původní hustotní data, jedná se tedy v podstatě o kontejner pro pohodlné uložení dat. Lze ji brát jako pomyslné 3D pole hustotních hodnot, s tím rozdílem, že textura navíc využívá jak nativní podporu grafických karet (GPU), tak některých metod dostupných pro textury, jako jsou třeba různé typy interpolací. Knihovnu se tedy dá zařadit do kategorie knihoven využívající textury k přímému vykreslování [48]. Knihovna podporuje všechny tři typy zobrazení, tj. DVR, MIP a zobrazení povrchů. Podporuje jak 1D, tak 2D transfer funkce, kde u 2D se přihlíží specificky k velikosti gradientu. Knihovna také podporuje řezy v datasetu a vykreslení již interpolovaných CT snímků. Je zde podpora pro medicínské formáty DICOM, NRRD, NifTi a případně další formáty jako klasický JPEG.

Existuje také tento zajímavý autorův blog [49], kde je více informací o knihovně. V blogu se řeší jednotlivé vizualizační problémy a celkové zakomponování řešení do Unity enginu. Blog obsahuje velmi kvalitní zpracování důležitých informací.

7.5.2 Export datasetů a segmentačních map z programu Slicer3D

Do aplikace je nutné lékařská data správně nahrát. Aplikace umí pracovat s konkrétními CT datasety, avšak lékaři mají převážně přístup k celým pacientovým studiím, které často obsahují více různých typů datasetu. Export vybraného CT datasetu ze studie je proto nutné provést v některém z již dostupných nástrojů. Nástroj, který se pro extrakci datasetů a následně i extrakci label map osvědčil, je Slicer3D. V příloze práce i na GitHubu je připraven vizuální návod popisující extrakci konkrétního datasetu z pacientovy studie.

Exportování **objemové** label mapy je potom zvláštní případ. Cíleně je zmíněno slovo objemová, protože label mapa má stejný voxelový formát jako konkrétní dataset, ale s tím rozdílem, že label mapa místo hustotních hodnot obsahuje označení jednotlivých segmentů. Label mapy mohou například segmenty označovat podle jejich ID, nebo mohou označovat minimální, průměrné či maximální hodnoty daného segmentu. Číselné hodnoty jednotlivých segmentů se v label mapě tedy můžou lišit, ale segment musí ve všech svých voxelech obsahovat stejné číslo, aby label mapa splňovala svůj účel a daly se segmenty správně oddělit.

Například label mapa, která obsahuje průměrné hodnoty segmentu, se může hodit přímo pro zobrazování. Při zobrazení této label mapy se ale ztratí hodně detailů, zejména pokud nejsou všechny důležité části segmentované. Avšak tento typ label mapy sloužil v prvotních verzích aplikace pro zobrazování, do té doby, než se do aplikace dodělala plná podpora pro segmentaci s využitím label map. O tom bude psáno v podsekci 7.5.7. Pokud jsou navíc všechny části segmentované, dá se tato label mapa v pořádku zobrazit bez ztráty větších detailů. Dobře to lze vidět například na Ircad datasetech, které jsou dostupné ze stránek [50] Ircadu.

Všechny label mapy by nicméně ve svých metadatech měly obsahovat i názvy jednotlivých segmentů. Zejména typ label mapy, který je použit v tomto projektu, by tyto názvy měl určitě obsahovat. Konkrétně o tom mluví Slicer dokumentace [51], kde se nachází i návod pro extrakci label mapy z programu Slicer3D. Bylo ale zjištěno, že takto extrahovaná label mapa potřebná metadata neobsahuje. Toto zjištění bylo ověřeno přímo v Unity při snaze přečíst a vypsat všechna metadata pomocí knihovny SimpleITK. V tomto případě soubor obsahoval jen pár úvodních hlaviček a zbytek metadat včetně jmen segmentů chyběl.

Po krátkém experimentování v programu Slicer3D však bylo zjištěno, že se z něj dá exportovat stejná label mapa jiným způsobem, která již tyto metadata obsahuje a která se dají přečíst. Pro správný postup extrakce label mapy je v příloze práce i na GitHubu znovu připraven vizuální návod. Pokud jsou v label mapě metadata o názvech segmentů obsažena, aplikace je umí přečíst a odpovídajícím způsobem je přiřadit k patřičnému segmentu. Pokud label mapa však tyto informace neobsahuje, funguje identifikace segmentů ve scéně jen podle barvy (která je stejná pro voxely stejného segmentu).

7.5.3 Načítání nových datasetů

Návod k přidání nových datasetů je zobrazen v této sekci manuálu. Aplikace byla vytvořena tak, aby dynamicky uměla pracovat s nově vytvořenými složkami v adresáři StreamingAssets. StreamingAssets adresář je specifický (citlivý na velikost písmen) Unity adresář, kde cesta k němu je interně vždy známá bez ohledu na platformu. Výhoda je pak v tom, že s tímto adresářem lze pracovat stejně při vývoji aplikace v editoru a také u sestavené aplikace. V tomto adresáři aplikace skenuje počet a obsahy složek. Při vytvoření nové složky pro dataset je její jméno zobrazeno v aplikaci, společně s volitelným náhledovým obrázkem.

Načítání základních dat k těmto datasetům probíhá již na začátku v třídě HandMenu.cs, která používá návrhový vzor Singleton. Tato třída obsahuje globální ovládací prvky ke všem datasetům. Hand menu prefab pak obsahuje všechny aplikaci dostupné datasety, které jsou reprezentovány třídou DatasetButton.cs, která obsahuje veškeré informace ke konkrétnímu datasetu. Tato třída v sobě obsahuje informaci o volumetrickém objektu ve scéně, náhledovou texturu datasetu, jeho jméno atd. Jednotlivé datasety jde do scény vložit dvojklikem jejich tlačítek, které se nachází v hand menu (tyto tlačítka lze také vidět na obrázku 7.7 nahoře). Dvojklik je použit z důvodu, že na zařízení Hololens 2 často docházelo k nechtěnému kliknutí a dataset se sám omylem vložil do scény. Počátečná verze fungovala s podržením tlačítka, což zabraňovalo nechtěnému kliknutí, ale dvojklik se ukázal jako lepší řešení. Vložení datasetu do scény je ukázáno v této sekci návodu. Posuvná lišta, kterou lze v návodu vidět, používá posuvnou komponentu, která je dostupná v základním MRTK repozitáři.

O konkrétní načítání potřebných dat pro vykreslovací objekt se stará třída VolumeDataControl.cs. Načítání probíhá asynchronně s maximálním využitím sekundárních vláken, aby nedocházelo k zamrzání celé aplikace. V původní verzi vykreslovací knihovny toto asynchronní načítání nebylo, takže se muselo implementovat. O tom ale bude více řečeno v podsekci 7.5.5.

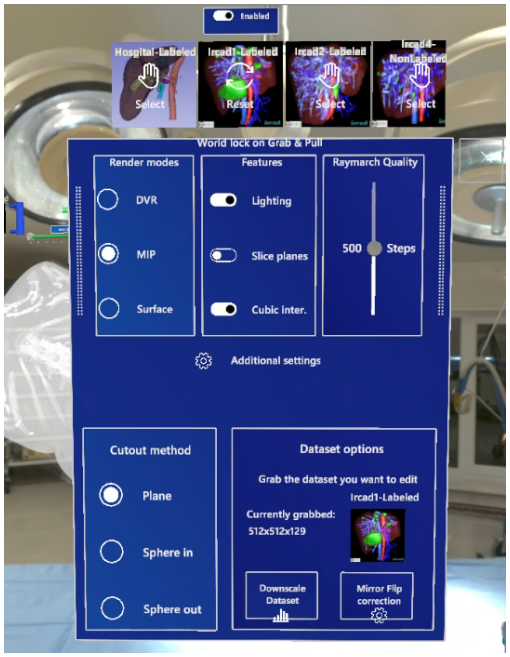
Načtené data jsou již v tzv. Hounsfieldových jednotkách (radiologické hustoty). O převod se automaticky stará knihovna SimpleITK, která je použita pro čtení dat z lékařských formátů a která u převodu aplikuje dostupné DICOM parametry RescaleSlope a RescaleIntercept [23]. Načtené hodnoty ze SimpleITK knihovny byly také srovnány s hodnotami z programu Slicer3D a hodnoty jsou stejné, což potvrzuje, že knihovna tento převod provádí automaticky. Při načítání dat je také brán v potaz parametr 0028|0030 [52], podle kterého se případně upravuje měřítko datasetu, neboť pixely mohou mít někdy odlišné tzv. „aspect ratio“.

7.5.4 Ovládaní aplikace

V aplikaci existují dva důležité ovládací systémy. Prvním je hand menu se systémem převážně globálního nastavování hodnot pro všechny datasety. Druhým je ovládací panel určený pro nastavování hodnot jednotlivých datasetů.

7.5.4.1 Menu systém

Globální nastavení datasetů je v aplikaci primárně řešeno pomocí tzv. hand menu [53]. Hand menu je specifická vlastnost MRTK knihovny určená především pro zařízení Hololens 2, kde k otevření takového menu dochází při detekci vnitřní dlaně. Pokud zařízení detekuje vnitřní dlaň, zakotví na danou pozici zvolený prefab. V aplikaci tento prefab obsahuje globální ovládací prvky datasetů a také možnost vložit je do scény, jak bylo zmíněno v minulé sekci. V menu se nachází volba vykreslovacího módu, nastavení kvality zobrazování a osvětlení, možnost zapnutí zobrazení řezů datasetu podle jednotlivých os a nastavení počtu kroků při RayMarchingu. Pro nastavení počtu kroků pro RayMarching byla do VolumeRenderingShader.shader přidána proměnná, která se aktualizuje v případě změny RayMarching posuvníku v hand menu. Hand menu také obsahuje sekci „Additional Settings“, kde se nachází méně využívané ovládací prvky, a sekce je rolovatelná. V této sekci se nachází výběr výřezové metody (bude vysvětleno dále v textu) a specifické ovládací prvky pro vybrané datasety. Hand menu i s rozšířenou sekcí lze vidět na obrázku 7.7.



Obrázek 7.7: Ukázka Hand-menu

Pro zachování soudržnosti byl podobný systém přidán i do VR verze, ve které detekce dlaně nefunguje kvůli použití ovladačů. Ve VR verzi se hand menu otevírá podržením primárního tlačítka na levém ovladači. Při podržení se menu zakotví k levému ovladači podobně jako s použitím Hololens 2. Toto menu se ve výchozím stavu otevírá dočasně a je otevřené, pokud Hololens 2 detekuje ruku, nebo pokud je ve VR zmáčknuté již zmíněné tlačítko. Pokud uživatel následně ruku před zařízením schová, nebo ve VR přestane držet dané tlačítko, menu se automaticky uzavře. Uživatel ale může menu i stabilně ukotvit v prostoru. K ukotvení stačí menu otevřít a poté ho uchopit druhou rukou,

kde při uchopení automaticky dojde k ukotvení menu v prostoru. Ukotvené menu lze pak zvětšit, otočit, přemístit či zavřít podle preferencí uživatele.

7.5.4.2 Konfigurační panel datasetu

Každý jednotlivý dataset po vložení do scény obsahuje důležité ovládací prvky, ve výchozím stavu zakotvené nad datasetem. Tyto konkrétní ovládací prvky v prostoru jsou spojeny s držátkem, které lze uchopit a celý panel přesunout podle preference uživatele. Mezi důležité ovládací prvky datasetu ve scéně patří hustotní interval, označující spodní a horní hranici zobrazovaných dat. Intervalový posuvník v MRTK knihovně není k dispozici, a proto bylo nutné ho vytvořit.

Implementace intervalového posuvníku využívá celkem tři jednoduché MRTK posuvníky. Dva z nich slouží k definici dolní a horní hranice intervalu, zatímco třetí posuvník slouží k jednoduchému posouvání obou hranic. Uživatel může uchopit hranice intervalu jednotlivě a posouvat dolní nebo horní hranici samostatně. Alternativně může uchopit střed intervalu, kde při jeho posunutí se obě hranice automaticky aktualizují. Pro tuto celkovou funkcionalitu byl vytvořen skript SliderIntervalUpdater.cs, který se stará o jednotlivé úkony intervalu. Oba posuvníky definující hranice intervalu spouští po změně hodnot *OnValueChanged()* Unity událost, která spouští tuto metodu 7.3.

```
public void OnChangeSliderValue()
{
    IntervalSliderValueChanged?.Invoke();

    float middleSliderSize=Mathf.Abs(_firstSlider.SliderValue-_secondSlider.SliderValue);

    Vector3 updatedScale = _middleSliderCollider.transform.localScale;
    updatedScale.x = middleSliderSize;
    _middleSliderCollider.transform.localScale= updatedScale;
    _middleSliderCollider.transform.position=
        (_firstSliderThumb.transform.position+_secondSliderThumb.transform.position)*0.5f;
}
```

Listing 7.3: Úprava středového posuvníku

IntervalSliderValueChanged je pak událost, na kterou je přihlášen skript **VolumeDataControl.cs**, aby dostával oznámení o změně intervalu. V metodě 7.3 se následně upravuje velikost a pozice středového posuvníku vzhledem k hraničním posuvníkům. Pro lepší přehlednost byla ke třetímu posuvníku přidána animace změny materiálu, aby uživatel viděl, že po ukázání na ovládací prvek jej může uchopit. Při uchopení středového posuvníku se spouští metoda 7.4, kde dochází k posunutí obou hraničních posuvníků, pokud je lze ještě někam posunout. Díky změně hodnot posuvníků dochází znovu ke spuštění metody 7.3 a data se ihned aktualizují do shaderu.

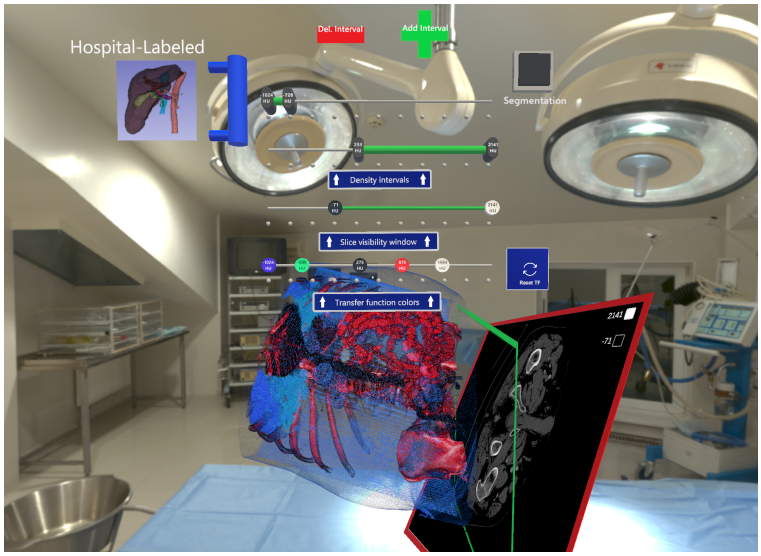
```
public void MiddleSliderUpdated(SliderEventData data) //Data stredoveho posuvniku
{
    float sliderRange=_firstSlider.SliderValue-_secondSlider.SliderValue;

    bool isSecondSliderGreater = sliderRange < 0;

    float shift = sliderRange * 0.5f;
    float lower = data.NewValue - shift;
    float upper= data.NewValue + shift;
    (float firstSliderNewValue, float secondSliderNewValue) = isSecondSliderGreater ? (lower, upper):(upper, lower);

    if ((firstSliderNewValue <= 1)&& (secondSliderNewValue <= 1))
    {
        if ((firstSliderNewValue >= 0)&& (secondSliderNewValue >= 0))
        {
            _firstSlider.SliderValue = firstSliderNewValue;
            _secondSlider.SliderValue = secondSliderNewValue;
        }
        else      //Pokud je hodnota nekterych z hranicnich posuvniku pod 0, posun oba slidery o stejny rozsah na start
        {
            float toFill = isSecondSliderGreater? _firstSlider.SliderValue: _secondSlider.SliderValue;

            _firstSlider.SliderValue -= toFill;
            _secondSlider.SliderValue -= toFill;
        }
    }
    else          //Pokud je hodnota nekterych z hranicnich posuvniku nad 1, posun oba slidery o stejny rozsah na konec
    {
        float toFill = 1 - (isSecondSliderGreater ? _secondSlider.SliderValue : _firstSlider.SliderValue);

        _firstSlider.SliderValue += toFill;
        _secondSlider.SliderValue += toFill;
    }
}
```

Listing 7.4: Posouvání středového posuvníku

Ve výchozím stavu je v datasetu jeden intervalový posuvník, ale lze jich postupně přidat více dle uživatelské preference. Princip funguje přes zaslání dvou polí float hodnot přímo do shaderu. Prvky prvního pole označují minimální hranice intervalů, druhé maximální. Shader neumí dynamická pole, proto jsou pole v shaderu inicializována na výchozí velikost pět set. Nepředpokládá se totiž, že by uživatel měl v jednu chvíli více než pět set posuvníků. Hodnoty v polích jsou ve výchozím stavu definovány na nulu. Pokud uživatel přidá nový posuvník, jeho minimální a maximální hranice se podle jeho indexu v obou polích aktualizuje a pole se následně pošlou do shaderu. Shader sice ví, že velikost obou polí je pět set, ale neví jaká část z toho rozsahu je aktivní, proto je také do shaderu zasílána informace, kolik existuje těchto aktivních intervalových posuvníků. Shader poté uvnitř kontroluje, zda hustota částice je v některém z intervalů. Pokud je hustota mimo všechny aktivní intervaly, dojde k zahození částice (ve fragment shaderu) a částice se stane kompletně průhlednou.

Do shaderu se vždy pošlou aktuální hodnoty polí, pokud dojde u jakéhokoliv posuvníku k zavolání metody 7.3, při které dojde k události *IntervalSliderValueChanged*. Jak již bylo zmíněno, skript **VolumeDataControl.cs** je na tuto událost u všech intervalů přihlášen a při jakékoliv změně pošle aktuální pole do shaderu.

Společně s těmito hustotními intervaly se v nastavovacím panelu nachází i sekce pro úpravu transfer funkce. Vzhledem k tomu, že jednotlivé datasety můžou mít různé intervaly Hounsfieldových hustot a pro lepší ovládání jsou normalizovány podle minimálních a maximálních hodnot, může každý dataset s výchozí transfer funkcí vypadat jinak. Uživatel může v transfer funkci pozice barev po speciálním posuvníku posouvat. Tyto barevné pozice byly názorně vidět na obrázku transfer funkce 5.3 a. Na obrázku 7.8 je vidět ukázka celkového konfiguračního panel datasetu a lze v něm vidět i tento barevný posuvník v AR/VR verzi. Barevný posuvník je spojen s hustotním intervalem a uživatel může přímo nastavit, jaká barva by měla být použita pro danou hustotu. Při takovémto přesunu barev efektivně dochází k vytvoření nové transfer funkce, která se následně pošle do shaderu. Uživatel by si měl takto primárně vystačit s nastavováním barevných pozic. Nastavování alfa hodnot u transfer funkce se již nestihlo, neboť se více pozornosti věnovalo segmentačnímu modulu, který obsahuje nastavování alfa hodnot. Tento segmentační modul by měl být v budoucnu primárně využit.



Obrázek 7.8: Ovládací prvky pro konkrétní dataset

Pokud uživatel z hand menu zapne tzv. slice mód, do datasetu se vloží rovinné snímky umístěné pro všechny tři osy v prostoru (jeden ze snímků lze také vidět na obrázku 7.8). Tento slice mód dovoluje uživateli zobrazit přímo CT snímky, tak jako by je viděl klasicky na monitoru. Se snímky lze ve scéně hýbat a různě je otáčet a zvětšovat. Při takovéto manipulaci dochází k interpolaci dat z původního datasetu na 2D snímek. Výchozí vykreslovací knihovna tuto funkcionalitu obsahovala, avšak snímky byly barveny podle zvolené transfer funkce. Dle zpětné vazby od chirurgů FNO bylo barvení snímku změněno na klasické černo bílé provedení, které je typické u prohlížení CT snímků

na monitoru. Kvůli tomu pro tyto snímky přibyl u konfiguračního panelu ovládací posuvník pro vhodnou volbu radiologického okna. Uživatel si tak může u snímků vylepšit kontrastní zobrazení pro zvolený dataset. Pro lepší představu, v této sekci návodu je manipulace s radiologickým oknem ukázána.

Pokud je u datasetu přítomná i label mapa, která nese informace o pozicích a rozsazích jednotlivých segmentů, zobrazí se v ovládacím panelu i možnost zapnutí segmentačního modulu. O segmentaci bude více povídat podsekce 7.5.7.

Na závěr této sekce, na GitHubu existuje celkový uživatelský manuál popisující ovládání a interakci s aplikací. Existuje i přímo manuál pro zdravotnické pracovníky, který je uzpůsoben pro potřeby nemocnice a jsou z něho odstraněny ostatní sekce, které jsou pro lékaře nepotřebné (na tyto manuály je v příloze práce pouze odkazováno, neboť kvůli své velikosti se do přílohy již nevešly).

7.5.5 Asynchronní načítání pro potřeby AR/VR

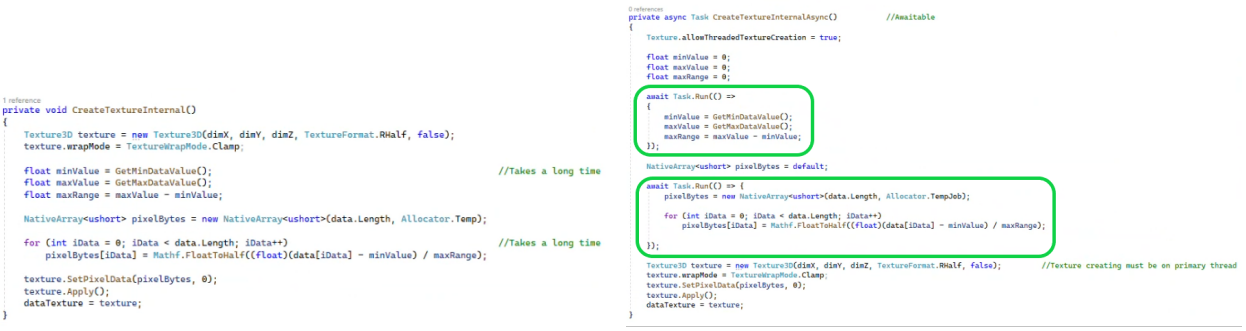
Využitá vykreslovací knihovna ve výchozím stavu prováděla veškeré operace na primárním vlákně, což způsobovalo problémy. Když dataset začal načítat a došlo k tvorbě potřebných textur a dalších časově náročných operací, aplikace se kompletně zasekla. Výchozí vykreslovací knihovna není primárně určena pro potřeby AR/VR a na monitoru to není takový problém. V případě AR/MR to také není problém, protože uživatel se normálně orientuje v prostoru a hologramy jsou jen dočasně vypnuté. Avšak ve VR je to velký problém, neboť dochází ke kompletní virtualizaci a v době načítání se zasekne úplně vše, včetně pozadí. Efekt je velmi nepříjemný a pravděpodobně by uživateli po nějakém čase způsoboval nevolnost. Načítání navíc není zrovna krátké, záleží na datasetu, ale pohybuje se v řádu desítek sekund až pár minut pro velké datasety. Bylo tedy nutné knihovnu upravit, aby pro náročné a dlouhé výpočty uměla využít jiného vlákna.

Pro toto řešení se obecně nabízí využití asynchronního programování [54]. Aplikace bude kód provádět klasickým způsobem, dokud se nedostane k výpočetně a časově náročné operaci. V tomto případě se následný blok kódu spustí v druhém vlákně. Na toto vlákno se pak počká, neboť je tzv. „awaitable“, a až vlákno výpočty dokončí, program se vrátí zpět k vykonávání. Takový přístup zabraňuje blokování hlavního vlákna, které se stará o hlavní operace v Unity, a aplikace běží bez zaseknutí.

Nevýhodou asynchronního programování je však nutnost převést potřebné metody na asynchronní varianty. Může se stát, že asynchronní část postupně začne patřičně zvětšovat velikost aplikace, protože je třeba mít pro více a více metod asynchronní variantu kvůli synchronizaci aplikace. Je tedy nutné vždy pečlivě vyhodnotit nejlepší možnou cestu, aby asynchronní variantu dostaly pouze metody, které to skutečně potřebují, a nemuseli bychom přepisovat celou aplikaci. Tento jev se někdy nazývá „async contagion“ nebo „zombie async“, kde asynchronní varianty začnou postupně infikovat celý zbytek kódu. Více je možné si o tom přečíst v dokumentaci [55]. Přesto

je v této aplikaci využití asynchronního programování velmi výhodné a knihovna byla s těmito asynchronními variantami upravena.

Při přepisování potřebných metod jsou náročné bloky kódu zabaleny do lambda výrazu *await Task.Run(()=>{[blok kódu]})*, kde daný blok se spouští s novým vláknem ze spravovaného fondu vláken. Ukázka přepsání takové metody se nachází na obrázku 7.9, kde je vidět srovnání stejné metody v synchronní a asynchronní verzi.



(a) Synchronní metoda

(b) Asynchronní metoda

Obrázek 7.9: Ukázka asynchronní metody (zeleně zvýrazněná části označuje asynchronní spuštění nových vláken z dostupného fondu vláken)

Přepisování není úplně snadné, neboť část kódu, která se přímo týká Unity, nelze přesunout do jiného vlákna, jinak dojde k vyjímce. Sem patří vytváření textur, operace s Transformem[2], jakékoliv aktualizace scény atd. Řešením je předem připravit potřebná data v druhém vlákně a teprve nakonec provést tyto specifické Unity činnosti v hlavním vlákně. Proto, jak lze vidět z obrázku 7.9 b, je i struktura kódu mírně upravena, aby byly specifické Unity činnosti pohromadě. Tvorba textur tedy může vést k lehkému zaseknutí, avšak toto zadržení je jen velmi krátké.

S dokončením asynchronních variant potřebných metod došlo zpětně také k přispění do původní knihovny pomocí pull requestu, neboť autor knihovny měl problém týkající se asynchronního načítání již otevřen. Asynchronní načítání je tedy nyní také dostupné u knihovny od původního autora. Po začlenění možnosti asynchronního načítání do původní knihovny se autor knihovny rozhodl na novém kódu již začít stavět a použít asynchronní načítání jako výchozí stav v knihovně. Autor zároveň dodělal základní podporu pro procentuální ukazatel načítání, kde aktualizace lze vidět na tomto pull requestu. Načítací ukazatel byl v našem projektu velmi výhodný, neboť načítání může být dlouhé a je dobré vidět, že se někam posouvá.

Základ kódu načítacího ukazatele byl od autora knihovny převzat, upraven pro účely AR/VR a následně bylo řešení začleněno do aplikace, kde u každého delšího načítání se zobrazují načítací etapy a procentuální stavy v nich. Jelikož vlákna na pozadí (backround threads) nemohou přímo komunikovat se scénou, je pro aktualizace použita speciální kolekce ConcurrentQueue. Jak název

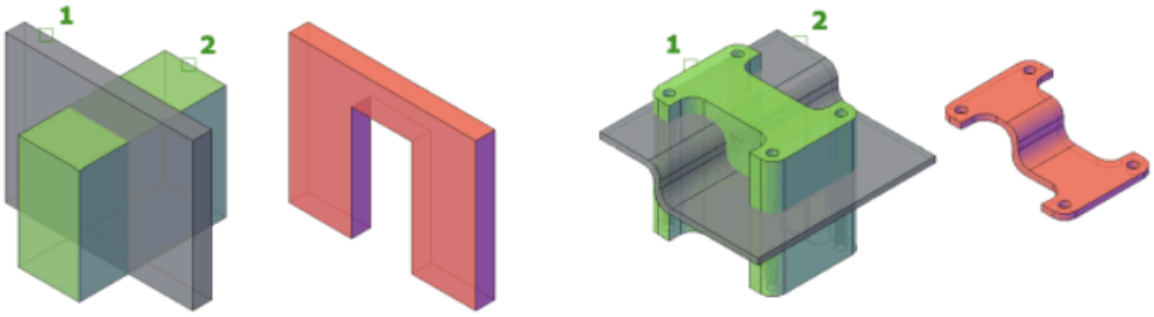
[2] Transform má v sobě uloženou pozici, rotaci a měřítko objektu ve scéně.

napovídá, jedná se o frontu, která má ošetřeny možné mezi-vláknové kolize. Do fronty se vkládají informace o stavu načítání z podřadných vláken a v aktualizační smyčce aplikace si hlavní vlákno z fronty tyto informace vyzvedává a patřičně aktualizuje vizuální elementy ve scéně. Aktualizace jsou řešeny na bázi celých procent, a mezistavy se přeskakují, aby zbytečně nedocházelo k zahlcení aktualizační fronty.

Je také dobré zmínit, že asynchronní varianty metod nejsou použity pouze s kombinací s vykreslovací knihovnou, ale i ostatní oblasti v aplikaci z asynchronního programování občas těží. Příkladem je například asynchronní ukládací systém, který automaticky běží na pozadí. O tomto systému bude více řečeno v podkapitole 7.5.8

7.5.6 Implementace sférického výřezu do DVR shaderu

Schopnost aplikace provádět řezy a výřezy nad daty chirurgové velmi ocenili a bylo nám řečeno, že je to pro ně velmi důležité. Knihovna ve výchozím stavu podporovala pouze rovinné řezy a výřezy pomocí krychle. Bylo však zmíněno, že místo krychle by pro ně bylo užitečnější provádět řezy pomocí koule. V této podsekci se mluví o tzv. výřezu, avšak pod tímto pojmem si v případě využití krychle, nebo koule lze představit booleovské operace rozdílu a průniku. Názorně je to vidět na obrázcích 7.10.



(a) Operace rozdílu

(b) Operace průniku

Obrázek 7.10: Znázornění výřezových metod využitých v aplikaci (převzato z [56],[57])

Vykreslovací knihovna však výřez pomocí koule ve výchozím stavu nepodporovala a bylo nutné ho dodělat. Autor knihovny pro výřezy používá transformační matice. Jedná se o local-to-local převodní matice, které jsou vytvořeny z výřezového a vykreslovaného objektu, tyto matice se pak posílají do shaderu. Ve fragment shaderu se poté zavolá speciální metoda, která přijímá parametr lokální pozici fragmentu a která uvnitř prozkoumá, jestli se fragment nenachází v některém z aktivních vyřezávacích objektů. Pokud je fragment v některém z těchto objektů, je zahozen a není brán v potaz. Metoda uvnitř pracuje s předpokladem, že střed objektu je nastaven lokálně na (0,0,0) pro ulehčení výpočtů. Vykreslovaný objekt má nicméně lokální rozsah od (0,0,0) do (1,1,1). Uvnitř

metody pro zjištění výřezu se tedy vždy fragment pozice posune o (-0.5,-0.5,-0.5), což efektivně způsobí, že střed vykreslovaného objektu se přesune do souřadnice (0,0,0). Uvnitř metody se následně lokální pozice fragmentu přenásobí transformační maticí výřezového objektu. Při takovémto přenásobení se pozice fragmentu dostane do lokální souřadné soustavy výřezového objektu a v této soustavě se pak celkově vyhodnotí, jestli pozice fragmentu leží či neleží ve výřezovém objektu. Tento postup je opakován pro všechny výřezové objekty. Rovinný řez funguje jednoduchým porovnáním osy Z, jak lze vidět v části kódu 7.5

```
bool isClipped = planeSpacePos.z > 0.0f;
```

Listing 7.5: Planární výřez

Výřez pomocí krychle poté kontroluje, jestli fragment je ve všech osách od středu vzdálen maximálně 0.5 délky, toto platí pro vnitřní řez, pro vnější je podmínka opačná. Tento systém funguje díky tomu, že střed objektu je nově v pozici (0,0,0) a objekt má lokální rozsah od (-0.5,-0.5,-0.5) do (0.5,0.5,0.5). Měřítko výřezového objektu je obsaženo v jeho transformační matici, výřezový objekt se tedy může zvětšovat/zmenšovat ve scéně a je to bráno v potaz. Nově implementovaný výřez pomocí koule pak kontroluje celkovou euklidovskou vzdálenost fragmentu od středu, což je vidět na kódu 7.6.

```
bool isClipped= length(sphereSpacePos) > 0.5; //Pro vnější výřez
bool isClipped= length(sphereSpacePos) < 0.5; //Pro vnitřní výřez
```

Listing 7.6: Výřez koule

Do knihovny bylo poté nutné dodělat specifický skript implementující rozhraní **CrossSectionObject**, upravit ho pro potřeby koule a vhodně ho poslat do **CrossSectionManager.cs** skriptu, který knihovna používá k posílání transformačních matic výřezových objektů do shaderu. Dodatečně vytvořený skript se poté jmenuje CrossSectionSphere.cs. Po dodělání sférického výřezu v projektu byl také otevřen tento pull request, který přidává sférický výřez do původní vykreslovací knihovny.

7.5.7 Segmentační modul

Výchozí verze vizualizační knihovny nepodporuje segmentaci. Nicméně možnost zobrazení segmentace byla pro chirurgy důležitá. Jak již bylo zmíněno, v budoucnu by měla být k dispozici neuronová síť, která bude provádět segmentaci nad původními daty automaticky. Momentálně máme k dispozici segmentaci pro jednoho pacienta, která byla provedena manuálně. Tato segmentace by měla být velmi podobná, jako z budoucího výstupu neuronové sítě. Manuálně segmentované data jsou v tomto stavu velmi důležitá, abychom si udělali celkový obraz toho, jak by měla data vypadat v budoucnu, a mohli podle toho program upravit. Délka provedení takové manuální segmentace je násobně delší než automatická segmentace na natrénované neuronové síti, a v budoucnu by měla být segmentační data díky neuronové sítí snadno dostupná. Do vizualizační knihovny bylo tedy třeba dodělat seg-

mentační podporu v podobě tzv. segmentačního modulu. Tento modul je dobrovolným dodatkem do knihovny a rozšiřuje její funkci, přičemž zachovává všechny předchozí možnosti knihovny.

7.5.7.1 Fungování segmentačního modulu

Po vložení datasetu do scény aplikace nejprve načte hustotní hodnoty z původního datasetu, které odpovídajícím způsobem zpracuje. Po tomto procesu se aplikace automaticky pokusí vyhledat label mapu tím, že proskenuje odpovídající složku, v níž by se měla nacházet. Pokud aplikace label mapu nenalezne, segmentační modul je ignorován. Avšak pokud je mapa nalezena, aplikace začne její data načítat a zpracovávat. Vzhledem k tomu, že formát NRRD bývá pro label mapy často používán a v programu Slicer3D představuje primární variantu, aplikace rovněž umožňuje z tohoto formátu extrahovat názvy a hodnoty segmentů z metadat, pokud jsou tyto metadata přítomná. Aplikace tedy postupně prochází veškerá NRRD metadata a snaží se získat všechny názvy a hodnoty segmentů, které následně ukládá do slovníku. Tento proces lze vidět na kódu 7.7.

```
ImageFileReader reader = new ImageFileReader();  //SimpleITK knihovna pro C#
string filePath="Path to the nrrd file";
reader.SetFileName(filePath);

int segmentNumber = 0;
List<string> metaDataKeys = reader.GetMetaDataKeys().ToList();
Dictionary<float,string> segmentInfos=new Dictionary<float,string>();

for(int i=0;i< metaDataKeys.Count;i++)  //Postupná extrakce všech názvu a hodnot segmentů
{
    string keyName = $"Segment{segmentNumber}_Name"; //V NRRD jsou klíče v tomto formátu
    string keyValue= $"Segment{segmentNumber}_LabelValue";
    if (metaDataKeys[i]==keyName)
    {
        float segmentValue = float.Parse(reader.GetMetaData(keyValue),CultureInfo.InvariantCulture);
        string segmentName = reader.GetMetaData(keyName);

        segmentInfos.Add(segmentValue,segmentName);
        segmentNumber++;
    }
}
```

Listing 7.7: Extrakce názvu segmentů z metadat

Načítání dat label mapy probíhá podobně jako u původního datasetu, přičemž jsou podporovány jednotlivé DICOM snímky i formáty jako NRRD a NIfTI. Data se postupně zpracovávají do float pole. Důvodem, proč zde není využito pole celých čísel, je skutečnost, že 3D textura v Unity podporuje pro skalární formát pouze uložení ve formátu single (float) a half precision. Toto je uvedeno v dokumentaci [58]. Byl vyzkoušen také R16 formát, který by měl uchovávat 16bitové celé číslo v prvním kanálu, avšak při následném spuštění aplikace, byla textura při prohlédnutí ve frame debuggeru zcela černá a extrakce hodnot nefungovala správně, přičemž konzole nezaznamenala žádnou chybu. Formáty Rhalf a Rfloat pro label mapu fungují bez problémů, a proto jsou zde využity.

Načtené hodnoty label mapy jsou převedeny na indexy v poli, které postupně označují čísla segmentů. Tímto způsobem jsou řešeny různé typy label map. Aplikaci tedy nezáleží, zda dostane label mapu s minimální, průměrnou či maximální hodnotou hustoty pro jednotlivé segmenty. Data se postupně projdou a vytvoří se slovník (C# používá pro Dictionary interně hash tabulku), do kterého jsou zapsány všechny unikátní hodnoty segmentů. Slovník se následně projde, vytvoří se pomocná kolekce, do které se vloží klíče ze slovníku a ty se poté seřadí podle velikosti. Do slovníku se jako hodnota k jednotlivým klíčům pak přiřadí již seřazená pozice. Tento proces je zobrazen na kódu 7.8.

```
private void OrderLabelDictionary(Dictionary<float,float> labelValues)
{
    List<float> orderedKeys = labelValues.Keys.OrderBy(x=>x).ToList(); //Seřazení klíčů

    for (int i = 0; i < orderedKeys.Count; i++)
        labelValues[orderedKeys[i]] = i;
}
```

Listing 7.8: Seřazení velikosti segmentů

Při následné tvorbě nativního pole textury se při zápisu pro každý prvek nahlédne do slovníku a do pole se přiřadí pozice segmentu z hodnoty, která je uložena společně s klíčem. Jak již bylo zmíněno, ideální by bylo využít celá čísla, avšak half a float jsou v tomto případě jediné fungující formáty. Využití slovníku je zde výhodné kvůli přístupové rychlosti k hodnotám podle klíče díky internímu využití hash tabulky.

Při tvorbě textury je poté nutné zvolit interpolaci k nejbližšímu sousedovi. Výchozí interpolace je trilineární, avšak z podstaty dat je nesmyslné využít lineární interpolaci, neboť si lze představit scénář, kde dva segmenty vedle sebe mají zcela rozdílné číslo segmentu, např. číslo jedna a číslo dvacet. Při lineární interpolaci by se mohlo na půli cesty dosáhnout čísla deset, přestože oba dva segmenty nemají se segmentem číslo deset nic společného a došlo by k chybnému přiřazení. Interpolace k nejbližšímu sousedovi sice znamená, že přechody budou více tzv. „zubaté“, nicméně zachovává správné chování.

Po vytvoření textury z label mapy dojde k jejímu poslání do shaderu. Hlavní RayMarching shader poté používá direktivu: **#pragma multi_compile LABELING_SUPPORT_ON**. Podle této direktivy se zapíná a vypíná logika segmentačního modulu v shaderu. Konkrétní kód v shaderu obsahující segmentační logiku je obalen výrazem **#ifdef LABELING_SUPPORT_ON [Kus kódu] #endif**. V shaderu sice lze použít klasické větvení, ale využití této direktivy znamená, že vzniknou dvě verze shaderu pro obě varianty podmínky a následně se použije verze shaderu podle toho, jaká klíčová slova jsou zapnuta. Tímto způsobem se obejde případné větvení v shaderu a dojde k úspoře výkonu. Tento typ direktivy se u shaderů většinou vyplatí používat a většina standardních shaderů v Unity obsahující možnosti volby přes shader property (editovatelné z editoru) ji také využívají. Nícméně je důležité zmínit i negativum, kde více shaderových variant znamená delší kompilaci a více zabrané paměti. Pro zapnutí či vypnutí klíčového slova stačí tyto informace předat

konkretnímu materiálu. V aplikaci se o zapínání segmentace stará metoda 7.9, která se nachází ve VolumeDataControl.cs

```
private void TurnMaterialLabelingKeyword(bool value)
{
    if (value) VolumeMesh.material.EnableKeyword("LABELING_SUPPORT_ON");
    else       VolumeMesh.material.DisableKeyword("LABELING_SUPPORT_ON");
}
```

Listing 7.9: Zapnutí segmentace

Do shaderu se společně s texturou posílá i pole barev, které obsahuje jak RGB složky, tak i alfa kanál. Pokud je segmentační modul zapnutý, na místě fragmentu se z textury podle interpolace k nejbližšímu sousedovi zjistí číslo segmentu, ze kterého se dá odečtením jedničky získat index pro přístup do pole barev[3]. Tento index se poté vloží do pole barev, kde jsou dostupné informace o barvě segmentu. Pro lepší vyhlazení hran bylo otestováno, že je vhodné přenásobit alfa kanál barvy segmentu hodnotou hustoty v daném místě (hustota se získává z druhé textury). Tímto způsobem se odstraní mnoho zubatých hran, protože orgány obvykle ztrácí hustotu na okrajích. Pole barev se do shaderu posílá z ovládacího panelu, kde uživatel může měnit alfa kanál segmentu a případně i RGB barvu. Ovládání alfa kanálu je odděleno od výběru barvy, protože se předpokládá, že operace s alfa kanálem budou primární a ke změně barvy bude docházet jen občas. Pro výběr barvy musí uživatel navíc rozkliknout barevné tlačítko označující náhledovou barvu segmentu.

Aplikace dynamicky vytváří odpovídající počet posuvníku podle počtu existujících segmentů, který získá podle počtu klíčů v již zmíněném slovníku. Na začátku je každému segmentu přiřazena počáteční barva, která vychází z posunu odstínu (hue), aby bylo dosaženo lepší odlišnosti mezi vytvořenými barvami. Metoda pro tvorbu odlišných barev je vidět na kódu 7.10. Jako parametr se jí předává celkový počet segmentů.

```
public Color[] CreateDistinctColors(int numberOfColors)
{
    Color[] colors = new Color[numberOfColors];
    float incrementStep = 1f / numberOfColors;

    for (int i = 0; i < numberOfColors; i++)
        colors[i] = Color.HSVToRGB(i * incrementStep, 1f, 1f);

    return colors;
}
```

Listing 7.10: Tvorba odlišných barev

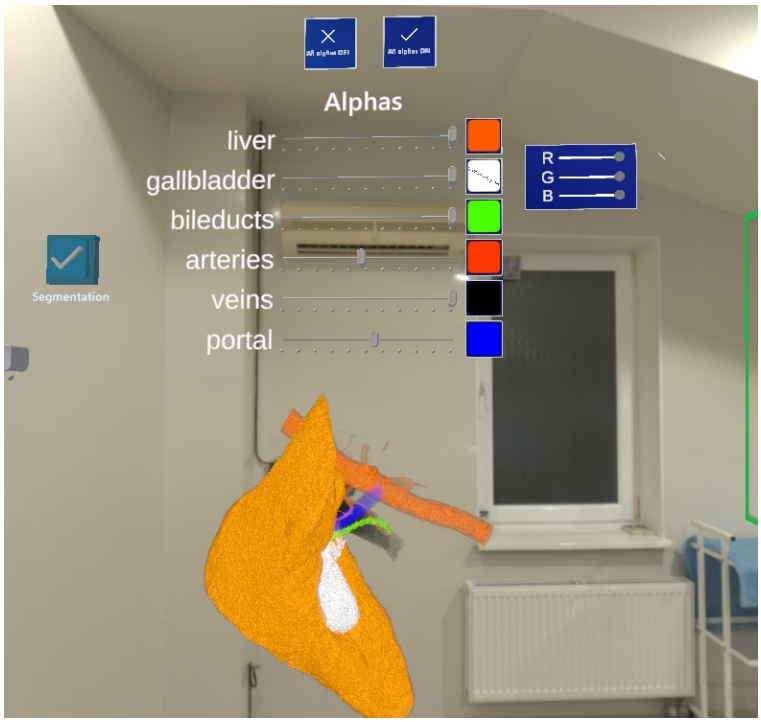
K tvorbě jednotlivých segmentů a jejich ovládacích prvků dochází ve skriptu **VolumeDataControl.cs**, a segmenty jsou reprezentovány třídou Segment.cs. Třída **Segment.cs** obsahuje událost pro

[3]Segmenty s číslem nula, znamenají, že se nejedná o žádný reálný segment. Nulové segmenty se tedy automaticky zahazují

změnu barvy, která se spouští při změně hodnoty alfa posuvníku nebo při novém výběru barvy. **VolumeDataControl.cs** je přihlášen na událost všech těchto segmentů, a pokud v nějakém segmentu dojde k jeho aktualizaci, pošle skript aktuální pole barev do shaderu.

Bylo třeba ještě vytvořit systém pro výběr barvy segmentu. Knihovna MRTK obsahuje experimentální color-picker, avšak tento ovládací prvek nefungoval dobře. Vybírání barev bylo zbytečně složité a prvek obsahoval chyby, kde specifické části nešlo ovládat ukazatelem, ale jen úchopem. Pro účely aplikace byl proto vytvořen nový jednoduchý modulární color-picker a třída CustomColorPicker.cs. Nový color-picker obsahuje tři posuvníky pro barevné složky a zavírací tlačítko, kterým se dá zavřít. Prvek se dá otevřít metodou, která jako parametr bere rozhraní **IColorPickerListener** (rozhraní obsahuje jednu metodu *OnColorUpdated()*), pozici a výchozí barvu. Třída segment toto rozhraní implementuje. Prvek funguje jako Singleton a ve scéně je pouze jeden. Pokud uživatel klikne na konkrétní barevné tlačítko u segmentu, segment zavolá otevření prvku, kde předá své hodnoty do parametrů. Pokud poté uživatel změní některou z barevných složek u color-pickeru, zavolá se metoda *OnColorUpdated()* z již uloženého rozhraní. Segment v případě této aktualizace vyvolá událost pro změnu barvy a přihlášený **VolumeDataControl.cs** pošle zmíněné nové pole barev do shaderu.

Celková interakce se segmentačním modulem je znázorněna v návodu v této sekci na GitHubu. Na obrázku 7.11 je pak vidět, jak tento segmentační modul vypadá. V segmentačním modulu jsou také pomocné tlačítka pro zapnutí/vypnutí všech segmentů, pokud by to uživatel potřeboval.



Obrázek 7.11: Segmentační modul

7.5.8 Systém ukládání

Během vývoje aplikace bylo navrhnuto, že by bylo vhodné ukládat konfigurační hodnoty pro jednotlivé datasety, které by poté šlo znovu načíst, aby uživatel nemusel při každém spuštění aplikace znovu nastavovat ovládací prvky. V aplikaci byl proto implementován asynchronní systém ukládání, který funguje automaticky bez nutnosti uživatele s ním jakkoliv manipulovat. Potřebné hodnoty jsou serializovány a uloženy ve formátu JSON. Pro JSON serializaci a deserializaci je použit NuGet balíček Newtonsoft.Json.

Nejprve bylo nutné připravit kontejnery pro ukládaná data. Kontejnery musely být vytvořeny, protože reálné třídy obsahují mnoho atributů, které jsou pro ukládání zbytečné. Některé třídy také obsahují kruhovou závislost, což je obecně pro JSON serializátor problém. Tyto kontejnery jsou zjednodušené verze reálných tříd a struktur, které obsahují pouze podstatné informace. Dané kontejnery jsou ve skriptu DatasetSaveData.cs. Skript Converters.cs pak obsahuje pomocné metody pro převod mezi těmito kontejnery a reálnými třídami a strukturami. Třída DatasetSaveSystem.cs poté obsahuje patřičné metody pro načtení uloženého souboru, aktualizaci reálných objektů na základě uložených hodnot a metodu pro uložení hodnot. **DatasetSaveSystem.cs** je pak použit jako komponenta ve třídě **VolumeDataControl.cs**, která systém využívá. Během načítání nového datasetu se z uložených dat postupně extrahují příslušné hodnoty.

Automatický systém ukládání funguje ve spolupráci s rozhraním IMixedRealityInputHandler, které je součástí knihovny MRTK. Toto rozhraní obsahuje dvě metody: *OnInputDown()* a *OnInputUp()*. Třída **VolumeDataControl.cs** toto rozhraní implementuje. Metoda *OnInputDown()* je prázdná, avšak metoda *OnInputUp()* volá asynchronní ukládací metodu z **DatasetSaveSystem.cs**. Rozhraní **IMixedRealityInputHandler** funguje pro většinu typů uživatelských interakcí s objekty ve scéně, s výjimkou stisknutí MRTK tlačítek. Při stisknutí specifických tlačítek tak dochází k dodatečnému zavolání ukládací metody. Tento „observer“ vzor zajišťuje, že hodnoty jsou uloženy až po ukončení uživatelské interakce s objekty. Tím je zabráněno zbytečnému zápisu na disk a zajistí se, že ukládané hodnoty se aktualizují až tehdy, pokud uživatel s objekty ve scéně manipuloval a je třeba provést uložení. Asynchronní ukládání společně s využitím fondu vláken pak zajišťuje, že diskové operace nejsou prováděny na primárním vlákně a nedochází tak k žádnému zastavení aplikace. Ukládání je proto naprosto plynulé a uživatelem nedetekovatelné.

Každý dataset pak obsahuje svůj vlastní ukládací soubor, který se nachází ve složce s názvem Saves u konkrétního datasetu. Pokud při vkládání datasetu do scény ukládací adresář nebo příslušný soubor neexistuje, vytvoří se nový. Změny jsou tak ukládány pro jednotlivé datasety a nejedná se o globální ukládání stavu scény. To znamená, že při manipulaci s objekty nedochází k velkému zápisu na disk. Aplikace ukládá celkový Transform (pozice, rotace a měřítko) datasetu, ovládacího panelu, výřezových objektů (mezi něž patří výřezová koule a rovina) a také jednotlivých snímku ve slice módu. V aplikaci se ukládá počet hustotních posuvníků, jejich intervalové hodnoty, transfer funkce s pozicemi jednotlivých barev a radiologické okno pro slice mód. Pokud je dostupný segmentační

modul, ukládají se také alfa posuvníky a celkové barvy jednotlivých segmentů.

7.6 Modelové řešení (Metoda nepřímého zobrazení)

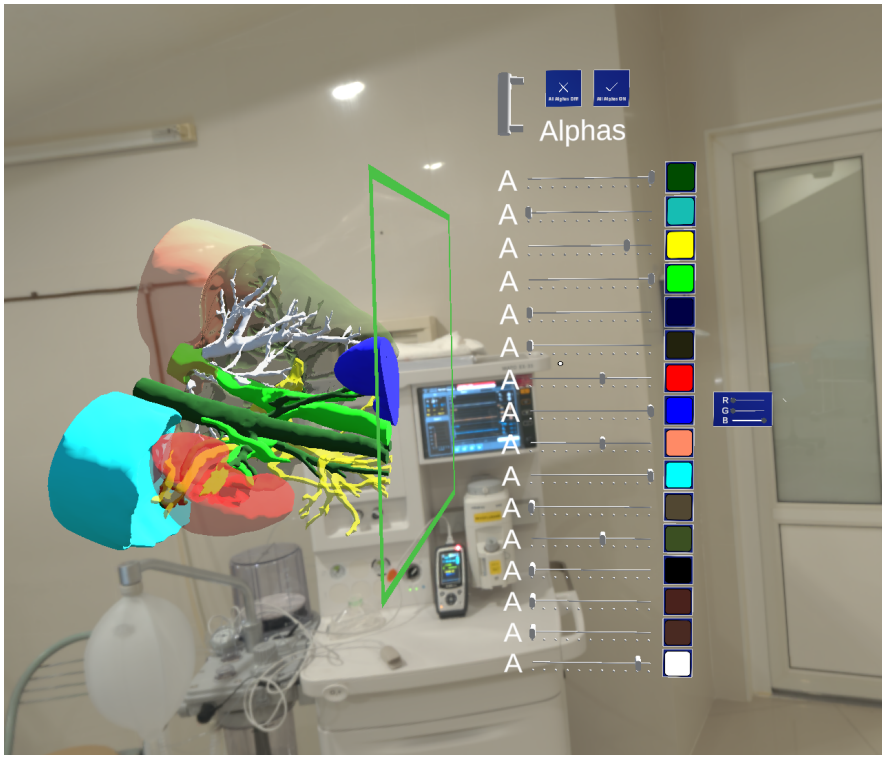
Vývoj aplikace se surface modely proběhl ve větvi Models v repozitáři projektu. Stejně jako předchozí volumetrická větev, i tato větev obsahuje README soubor, který poskytuje důležité informace pro zprovoznění projektu. Chronologicky je tato větev nejstarší, protože vývoj s modely začal nejdříve a větev navazuje na vývoj ze sekce 7.2. Tuto větev však postupně nahradila volumetrická větev, a ačkoliv je řešení na modelové větvi funkční, vývoj zde byl velmi upozaděn kvůli prioritizaci volumetrické větve. Modelové řešení tedy není v úplně uživatelsky přívětivém formátu. Ve srovnání s volumetrickou větví zde také některé prvky, jako například hand menu nebo ukládací systém chybí, nebo se nestihly dodělat.

U volumetrické větve je možné přidávat nové datasety do odpovídajících složek v již sestaveném řešení, což neplatí pro modelové řešení. V případě potřeby zobrazení dat nového datasetu, je v současné verzi projektu nutné patřičný model z datasetu vytvořit, zpracovat ho, vyčistit a poté sestavit novou verzi aplikace v Unity. Modelové řešení má obecně více kroků, a oproti volumetrické verzi, kde lze data v podstatě jen vložit do složky a hned je zobrazit, je zde proces relativně složitý[4]. Automatizace zpracování modelu, která by s touto složitostí pomohla, se již nestihla udělat, neboť větev měla velmi malou prioritu. Je důležité zmínit, že modelové řešení vyžaduje již hotovou label mapu. Pokud taková mapa pro dataset není dostupná, nelze modelové řešení použít. Tato label mapa se například v programu Slicer3D dá použít pro extrakci modelu v OBJ formátu se zachováním barev ze segmentace. Celý proces tvorby OBJ modelu ze segmentace, následnou úpravu modelu v Blenderu a pak importování a krátkou ukázku v Unity obsahuje tento zpracovaný video návod, který je také dostupný v README v sekci o zpracování modelu.

Modelové řešení využívá stejnou knihovnu Open Fracture pro řezání modelů v reálném čase, která byla použita v sekci 7.2. Knihovna je zde upravená tak, aby umožňovala podobné chování rovinných řezů, jako ve volumetrické větvi. Model se proskenuje pro existující segmenty a v kódu se k nim dynamicky podle jejich počtu vytvoří ovládací prvky. Ke každému segmentu se z kódu také automaticky přidá vhodně nastavený skript z OpenFracture knihovny, umožnující řezání modelu. Řezání modelu nefunguje kontinuálně, jako ve volumetrické větvi, a to z výkonnostních důvodů. Pokud uživatel rovinu uchopí a následně jí pustí, po puštění se mesh síť začne automaticky řezat. Řezání probíhá ve všech segmentech separátně, neboť jsou brány jako samostatné objekty, ale ve výsledku proces vypadá jako jeden velký řez v daném místě. Jak již bylo zmíněno v sekci 7.2, díry v jednotlivých řezech jsou automaticky zaplněny novými vrcholy se stejným materiálem jako obsahuje segment.

[4]Poznámka na okraj: teoreticky by bylo možné mít v aplikaci v reálném čase modelovou rekonstrukci dat nad původními daty, například pomocí algoritmu marching cubes. Avšak náročnost na výkon u takového algoritmu v reálném čase by narostla natolik, že by se v podstatě odstranily zbývající výhody modelového řešení, což je primárně možnost běžet samostatně na zařízeních bez nutnosti externího výpočetního výkonu.

Stejně jako ve volumetrické větvi v segmentačním modulu, funguje zde ovládání primárně pomocí alfa posuvníků, které nastavují průhlednost jednotlivých segmentů. Barvy jednotlivých segmentů jsou ve výchozím stavu extrahovány z dodaného FBX modelu (proces vytvoření FBX modelu je vidět na již zmíněném video návodu), který obsahuje výchozí materiály pro jednotlivé segmenty. Barvy jednotlivých segmentů lze měnit podobně jako ve volumetrické verzi aplikace, kde výběr barev probíhá s již dříve vytvořeným modulárním color-pickerem. Na rozdíl od volumetrické větve, kde se barvy segmentů posílají přes pole do shaderu, zde dochází u jednotlivých segmentů přímo ke změně barvy materiálu v MeshRendereru. Oproti volumetrické větví zde nejsou názvy segmentů, protože data vycházejí z FBX modelu a ne z lékařských dat, které by tyto informace obsahovaly. Identifikace segmentů tak probíhá výhradně podle barvy. Ukázka modelového řešení je zobrazena na obrázku 7.12



Obrázek 7.12: Ukázka aplikace s modely

Hlavní výhodou modelového řešení je, že aplikace může po sestavení fungovat na samostatných zařízeních bez nutnosti externího výpočetního výkonu. Aplikace byla testována na Hololens 2 a Quest 2 bez připojení k počítači. Na druhou stranu, nevýhodou je, že modely jsou založeny na label mapě a segmenty tak ztrácí často důležité detaily, které by se nacházely uvnitř segmentu. Zatímco volumetrické vykreslování zachovává tyto informace, modelové řešení se soustředí především na vrcholy modelu, které tvoří daný povrch a ne na vnitřní detaily segmentů.

Během vývoje modelové větve došlo k tzv. shader stripping problému. Tento problém se proje-

voval pouze v sestavené aplikaci, zatímco v editoru nebyl přítomen. Kvůli tomu byl problém objeven až v pozdější fázi vývoje, při testování sestavené verze aplikace. Tato zkušenost zdůrazňuje nutnost provádět testy na již sestavených aplikacích, kde se mohou objevit dosud nepozorované problémy. Shader stripping spočívá v odstraňování nadbytečných variant shaderů, které nejsou v aplikaci používány, s cílem snížit velikost sestavené aplikace. Tento proces probíhá automaticky, když engine během sestavování zkoumá, zda nějaká část aplikace odkazuje na konkrétní variantu shaderu. Pokud není nalezena žádná reference, daná varianta shaderu se vynechá a v sestavené aplikaci již není přítomna. Obvykle jde o pozitivní jev, avšak v případě modelové aplikace způsoboval problémy. Aplikace za účelem úspory výkonu přepíná varianty shaderu přímo ze skriptu, a pokud to není potřeba, může k ušetření výkonu přepnout určitý segment do neprůhledné varianty shaderu (průhledné shadery jsou náročnější na výkon). Přepínání variant shaderů z kódu však způsobovalo problémy. Při spuštění aplikace jsou totiž všechny segmenty neprůhledné a až následně se pomocí posuvníků mění alfa kanál a varianta shaderu. Unity však při sestavení automaticky vynechalo průhledný shader, protože všechny materiály byly původně neprůhledné. Tento problém byl vyřešen přidáním nové ShaderVariant kolekce do nastavení Unity (**Edit -> Project settings -> Graphics -> Preloaded Shaders**). Kolekce byla nastavena, aby obsahovala potřebnou průhlednou variantu standardního shaderu. V sestavené aplikaci pak díky tomu nedojde k nechtěnému strippingu.

7.7 Multiplatformnost

Velkou výhodou současných enginů, jako jsou Unity nebo Unreal, je možnost sestavit aplikaci pro více platforem s minimálními změnami zdrojového kódu. V rámci této práce je konkrétně využita knihovna MRTK, která je primárně určena pro brýle Hololens 2. Nicméně tato knihovna podporuje také standard OpenXR, což zaručuje, že aplikace by měla fungovat relativně dobře i na ostatních XR zařízeních, možná jen s drobnými odchylkami, které lze opravit.

Vývoj aplikace byl původně zaměřen pouze na Hololens 2. Avšak díky multiplatformním možnostem vznikla během vývoje i verze pro virtuální realitu na zařízení Oculus Quest 2. Tato verze se nakonec uchytila nejvíce a je v době psaní této práce v nemocnici testována lékaři. Přechod z MR do VR byl relativně snadný, a aplikace se liší pouze v několika drobných aspektech. Pro tyto drobné odlišnosti existuje ve scéně nastavovací skript s názvem PlatformSpecific.cs, s jehož pomocí lze přímo v editoru pomocí enumu nastavit cílovou platformu. Tento skript aktivuje specifické funkce pro vybranou platformu. Například u Hololens 2 dochází k vypnutí pozadí, zapnutí QR rozpoznávání a zpřístupnění nastavovacího remoting prefabu. U Quest 2 se naopak nastavuje pozadí (v aplikaci je použito typické nemocniční prostředí), zapíná se specifická verze hand menu a aktivuje se skript pro sledování vstupů na VR ovladačích.

Skript **PlatformSpecific.cs** funguje ve scéně jako Singleton a při běhu aplikace si některé skripty z něj vytahují informace o aktuální platformě, podle kterých se rozhodují, zda mají zobrazit některé platformně specifické prvky. Například v hand menu, pokud se jedná o platformu VR,

jsou vypnuty a skryty některé ovládací tlačítka týkající se QR kódu, protože QR kód není ve VR použit. Některé funkční prvky jsou zase vypnuté u Hololens 2, jako například možnost měnit pozadí aplikace.

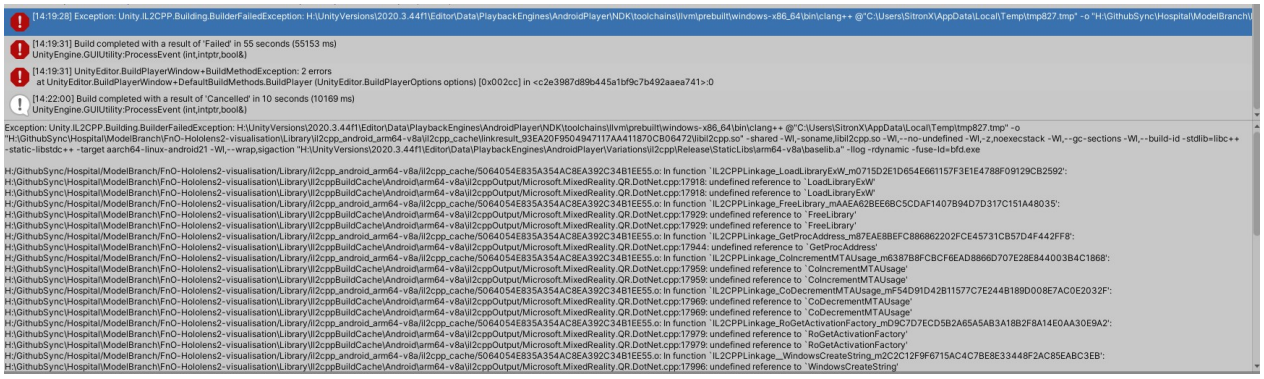
Důležitým aspektem při práci s různými platformami je správné nastavení typu XR inicializace, které se liší v závislosti na způsobu připojení zařízení. U Quest 2 zařízení dochází k připojení pomocí Link kabelu, nebo Air Linku přes Wi-Fi, přičemž v době spuštění aplikace je zařízení již připojeno k počítači. Naopak u Hololens 2 je nejprve spuštěna aplikace a až poté se provede připojení zařízení v rámci aplikace. V případě Quest 2 tedy probíhá XR inicializace ihned při spuštění aplikace, zatímco u Hololens 2 je nutná manuální XR inicializace po připojení zařízení. Toto manuální řešení pro Hololens 2 není nutné vytvářet od základu, neboť již existuje veřejný repozitář se vzorovým projektem. Z projektu stačí prefaby a potřebné skripty související s manuální inicializací pouze převzít a použít je v aplikaci. Typ XR inicializace - ať už automatická nebo manuální - se pak nastavuje přímo v konfiguraci projektu v Unity. U Hololens 2 je navíc nutné povolit možnost remotingu v nastavení projektu. Veškeré platformově specifické odlišnosti jsou pak uvedeny v této sekci dokumentace na GitHubu. Ačkoliv to nebylo otestováno, aplikace by měla fungovat i na dalších PC-VR zařízeních, pokud je sestavena z volumetrické větve (Quest 2 byl primárně testován). Jediný rozdíl u ostatních zařízeních by mohl být v otevírání menu, kde tlačítko pro otevření by se mohlo nacházet jinde než u Quest 2.

V práci dochází k využití celkem třech platforem. Volumetrická větev je určena pro stolní verzi Windows (Windows desktop standalone) kvůli využití PC hardwaru. Modelová větev je pak sestavena pro jednotlivá zařízení - u Hololens 2 se jedná o Universal Windows Platform (UWP) a u Quest 2 o platformu Android.

Při sestavování aplikace pro UWP je nutné nejprve zkompilovat Unity projekt do Visual Studio projektu, který slouží k následnému sestavení. Postup sestavení ve Visual Studiu je popsán v Microsoft dokumentaci. V exportovaném Visual Studio projektu obvykle není třeba provádět žádné změny. Projekt stačí otevřít, nastavit sestavovací parametry pro ARM64 Release a aplikaci pak sestavit a odeslat do zařízení přes USB-C nebo Wi-Fi.

Proces sestavování pro Android nevyžaduje žádný mezikrok, a APK balíček vytvořený při sestavení je nahrán na připojené zařízení přímo z Unity. U sestavování pro Quest 2 (Android) nicméně nastal problém u IL2CPP scripting backendu. Unity nabízí dvě možnosti sestavení, které lze vybrat z nastavení. Prvním je klasické Mono, které využívá kompilátor „Just in time“ (JIT), kde kód se kompiluje v reálném čase při běhu programu. Druhou možností je IL2CPP, které využívá kompilátor „Ahead of time“ (AOT), kde se kód kompiluje již při sestavení aplikace. V tomto případě je C# kód překládán do C++ během sestavení a výsledný binární soubor je pak sestaven z tohoto C++ kódu. Jak celkový překlad funguje je pak uvedeno v IL2CPP dokumentaci. Obě možnosti mají své výhody a nevýhody. Mono se vyznačuje rychlou dobou sestavení aplikace a v případě výjimek a chyb v aplikaci jsou pak chybové zprávy detailní a přesné. Mono je tak vhodné pro vývojářské testování. IL2CPP nadruhou stranu poskytuje lepší výkon a menší velikost sestavené aplikace. Další výhodou

pak je podpora většího spektra zařízení a knihoven. IL2CPP je tak vhodný pro Release verze aplikací. Avšak při sestavování pro Android pomocí IL2CPP došlo vždy k chybě, jak je znázorněno na obrázku 7.13.



Obrázek 7.13: Chyba pro sestavení IL2CPP na Androidu

Důvodem problémů se sestavením pro Android jsou pravděpodobně nějaké chybějící knihovny (týkající se QR rozpoznávání), které nemůže linker[5] u IL2CPP sestavení dohledat, ačkoliv QR rozpoznávání není ani pro Android verzi použito (je k dispozici pouze pro Hololens 2). Scripting backend byl proto změněn na Mono. Nicméně bylo zase zjištěno, že OpenXR podporuje pouze IL2CPP sestavení a ne Mono. Pro Quest 2 byl tedy změněn XR poskytovatel z OpenXR na OculusXR, který podporuje oba dva typy backendu. Sestavení s IL2CPP u něho také vykazuje chybu, ale aplikaci lze již sestavit pro Mono. Tato Android verze aplikace je tedy jediná, která využívá specifického poskytovatele OculusXR, zatímco ostatní verze používají OpenXR.

[5]Linker při sestavování aplikace tzv. linkuje různé moduly aplikace do jednoho souboru. V případě využití dynamických knihoven také řeší vhodné ukazatele na tyto knihovny.

Kapitola 8

Výkonnostní testování

V rámci této práce byly vyvinuty dvě aplikace, které se liší ve způsobu vykreslování. První aplikace využívá přímou vykreslovací metodu s RayMarchingem nad DICOM daty, což je extrémně náročné na výpočetní výkon. Bylo tedy důležité provést výkonnostní testování, aby bylo zjištěno, jak silný hardware je potřeba pro provoz takové aplikace. Výkonnostní testování druhé verze aplikace (s modely) je v této sekci rovněž obsaženo.

Měření výkonu na AR a VR zařízeních je problematické, protože jak Hololens 2, tak Quest 2 obsahují pevně nastavenou interní verzi V-syncu[1], což omezuje běžící aplikace na hranice stanovené zařízením. Tento interní V-sync se nepodařilo řádně vypnout. Byly vyzkoušeny možnosti jako nastavení „Target framerate“ na vysokou hodnotu, kompletní vypnutí V-syncu v Unity, správné nastavení Nvidia panelu a zakázání V-syncu. V případě remotingu byla vyzkoušena i změna nastavení Oculus aplikace pro AirLink, avšak ta pevně limituje snímkovací frekvenci na 60,72,90 a 120 snímků za sekundu. Zvláště problematické je, že kvůli tomuto V-syncu se aplikace při nedosažení cílových snímků za sekundu (FPS) často uzamkne na předem určených násobcích a snaží se využít funkcí jako je například Asynchronous Spacewarp 2.0, což je Oculus verze reprojekce. Tato metoda vkládá uživateli nové umělé snímky pro dosažení plynulosti aplikace (pro predikci snímku se berou v potaz pohybové vektory pixelů, optický tok a další [59]). I přes vypnutí této funkce měla aplikace tendenci se ve snímkovací frekvenci uzamykat na před-určené hodnoty. Tato interní starostlivost o snímkovací frekvenci je z uživatelského pohledu výhodná, protože umožňuje plynulejší a stabilnější zobrazování s omezením tzv. „frame-dropů“. Na druhou stranu, z pozice testera je tato situace velmi obtížná, protože výkon se takto hůře hodnotí a FPS hodnoty poskytují menší informační hodnotu. Hololens 2 toto zamykání FPS na předurčených násobcích neobsahuje, nicméně snímkovací frekvence je stále limitována maximální hodnotou 60 FPS.

Měření výkonu proběhlo následujícím způsobem: Pokud byla dosažena stabilní maximální snímkovací frekvence, je v tabulce uvedena hodnota **Max** pro dané pole (v závorce obsahující informaci o FPS), což nevypovídá jakých FPS hodnot by aplikace mohla dosáhnout, avšak označuje, že uži-

[1] V-sync je metoda pro synchronizaci snímkovací frekvence s obnovovací frekvencí monitoru zařízení.

vatelský zážitek je v takovém případě maximálně plynulý. Pokud FPS v průběhu běhu aplikace klesaly s ohledem na maximální FPS limit zařízení, je v tabulce uvedena i průměrná hodnota této frekvence. Označení **X** v tabulce znamená, že dle výsledků posledního měření o řádek výše nebylo další měření výkonu považováno za smysluplné.

Scénář měření zůstal pro obě verze aplikace stejný. Měření probíhalo vždy v sestavené aplikaci na cílovém zařízení, přičemž byly srovnány dva zobrazovací zařízení - Hololens 2 a Oculus Quest 2. Po spuštění sestavené aplikace byl v průběhu dvou minut výkon měřen. Na konci měření byla vypočítána průměrná hodnota FPS. Vzhledem k tomu, že aplikace neobsahuje žádné náhodné události typické pro hry, je snímkovací frekvence v průběhu provozu aplikace stabilní a nevyskytují se zde výrazné výkyvy. Při různých nastaveních a úhlech pohledu na konkrétní části datasetu může docházet k celkovému snížení výkonu, avšak i takové snížení je stabilní a výkon neprochází razantními změnami. Během měření bylo dbáno na to, aby byly podmínky pro všechna měření co nejpodobnější. Průměrná hodnota počtu snímků je tak smysluplnou hodnotou, která je uvedena v tabulkách.

8.1 Měření výkonu pro metodu přímého vykreslování

Pro měření výkonu první aplikace byly použity dva lékařské datasety. První pochází z Ircadu a je ke stažení dostupný na tomto odkaze. Druhý dataset je poskytnut přímo z FNO. Výsledky měření první aplikace využívající metodu přímého vykreslování jsou zobrazeny v tabulce 8.1.

Dataset	**Device**	**FPS Results (avg.)**		
		Native GPU	**GTX 1050**	**RTX 2060**
Ircad dataset 64x64y17z (69 632 vox.)	Quest 2	31.6 fps	Max (120)	Max (120)
	Hololens 2	6.7 fps	Max (60)	Max (60)
Ircad dataset 128x128y33z (540 672 vox.)	Quest 2	27.7 fps	Max (120)	Max (120)
	Hololens 2	5.9 fps	Max (60)	Max (60)
Ircad dataset 256x256y65z (4 259 840 vox.)	Quest 2	14.1 fps	81.6 fps	Max (120)
	Hololens 2	4.9 fps	Max (60)	Max (60)
Ircad dataset 512x512y129z (33 816 576 vox.)	Quest 2	3.2 fps	64.2 fps	Max (120)
	Hololens 2	3.1 fps	Max (60)	Max (60)
Nemocniční dataset 512x512y730z (191 365 120 vox.)	Quest 2	X	24.8 fps	85.9 fps
	Hololens 2	X	31.7 fps	Max (60)

Tabulka 8.1: Výsledky výkonnostního testování pro přímou vykreslovací metodu

Měření pro tuto aplikaci bylo provedeno jak na stolním hardwaru, tak i na samotných zařízeních (měření na zařízeních jsou v tabulce ve sloupci **Native GPU**). Z výsledků je patrný obrovský rozdíl ve srovnání s použitím dedikované grafické karty. Vzhledem k povaze vykreslování je ve volumetrické verzi aplikace maximálně zatížená grafická jednotka, kde v shaderech probíhá RayMarching. První sloupec tabulky vždy označuje dataset a jeho rozměr. Osy **x** a **y** udávají rozměr CT snímku, zatímco osa **z** značí počet takových snímků v datasetu. V závorce je pak uveden celkový počet voxelů v datasetu. Z důvodů testování byl dataset downsamplován, aby bylo demonstrováno, jaký vliv má počet voxelů na výkon aplikace.

Z výsledků vyplývá, že pokud aplikace běžela pro stejný dataset na samotném zařízení, dosahoval Oculus Quest 2, přestože má větší rozlišení než Hololens 2, typicky násobně lepšího výkonu. Důvodem je výkonnější XR2 čip. Z tabulky 8.1 je také patrné, že Quest 2 by teoreticky dokázal samostatně zobrazit nejmenší dataset na zhruba 30 snímků za sekundu. Nicméně tento dataset je tak extrémně redukovaný, že je prakticky nepoužitelný, protože ztratil většinu detailů a slouží pouze pro testovací účely. Pokud bychom srovnali oba zařízení s využitím PC hardwaru, Hololens 2 dosahuje vyšší snímkovací frekvence, právě díky nižšímu rozlišení. Z výsledků je zřejmé, že dedikovaná grafická karta GTX 1050 zvládá jen určitou velikost datasetů a v případě větších datasetů, což je i reálný případ datasetu z nemocnice, má i tato grafická karta výkonnostní problém. RTX 2060 se poté ukázala jako plně dostačující pro takový dataset, kde plynulost zobrazování byla dostatečná pro oba typy zařízení.

Vzhledem k možnosti existence větších datasetů, byla do aplikace přidána funkce pro downsampling datasetu. Tato možnost umožňuje efektivně zpracovat i velké a detailní datasety, přičemž downsampling ušetří výkon s minimální ztrátou kvality (platí pro velké datasety). Instrukce pro downsampling jsou uvedeny v návodu v sekci downsamplingu, kde je názorně ukázán postup.

8.2 Měření výkonu pro nepřímou vykreslovací metodu (modely)

Ve druhé aplikaci se pro vykreslování využívají klasické modely. V sekci 5.1 bylo zmíněno, že při exportu segmentačního modelu je vhodné provést redukci vrcholů, protože exportovaný model z label mapy obsahuje často zbytečný počet vrcholů. Výkonnostní testování druhé aplikace proběhlo zejména s testováním různých počtu vrcholů modelu. Pro testování byl použit stejný Ircad dataset z předchozího měření, avšak tentokrát byl z jeho label mapy exportován odpovídající model. Výsledky lze nalézt v tabulce 8.2. Testování zde bylo provedeno pouze na samotných zařízeních.

Model vertex multiplier	**Device**	**FPS Results (avg.)**
		Native GPU
0.05 mult. (161 400 vert.)	Quest 2	Max (72)
	Hololens 2	44.4 fps
0.1 mult. (322 800 vert.)	Quest 2	Max (72)
	Hololens 2	33.5 fps
0.2 mult. (645 600 vert.)	Quest 2	55 fps
	Hololens 2	22.6 fps
0.3 mult. (968 400 vert.)	Quest 2	40.1 fps
	Hololens 2	15.1 fps
1.0 mult. (3 228 000 vert.)	Quest 2	8 fps
	Hololens 2	5 fps

Tabulka 8.2: Výsledky výkonnostního testování pro nepřímou vykreslovací metodu

Při vývoji aplikace je nejdůležitější řídit se doporučeními od konkrétního výrobce zařízení, který maximální doporučený počet vykreslovaných vrcholů většinou uvádí. Pro Hololens je tento limit

okolo 100 000 trojúhelníků, zatímco pro Quest 2 je limit zhruba 750 000 trojúhelníků. Pokud limit není přesažen, vykreslování by nemělo představovat problém. Je třeba také zohlednit, že v aplikaci je pro každý segment vytvořen nový objekt, a tyto objekty mají různé materiály. V aplikaci lze pak upravovat průhlednost těchto jednotlivých segmentů, nicméně průhledné/poloprůhledné shadery mají vyšší výkonnostní nároky, než klasický neprůhledný shader. Počet vrcholů tedy není jediným ukazatelem výkonu a je třeba brát v potaz i tyto a další faktory. Z výsledků měření vyplývá, že Quest 2 dosahuje obecně vyššího počtu FPS než Hololens 2. Při vhodné úpravě modelu však lze dosáhnou kvalitní snímkovací frekvence na obou zařízeních.

Na závěr této sekce je třeba zmínit, že při testování na Quest 2 standalone (Android) se narazilo na zajímavou skutečnost, že v případě využití Vulcanu na Androidu, dosahovala aplikace výrazně nižší snímkovací frekvence než s použitím OpenGLES3. Na internetu lze nalézt více příspěvků týkajících se daného problému. Problém pravděpodobně nějak souvisí se změnou ve využití hloubkové textury u Vulcanu. Toto zjištění přináší důležitou poznámku, že výkonnostní problémy nemusí být vždy spojeny pouze s konkrétní aplikací, ale také třeba s grafickým API, které si se specifickým zařízením nemusí úplně rozumět. V případě výkonnostních problémů je proto vždy vhodné vyzkoušet více variant.

Kapitola 9

Nasazení aplikace v FNO a zpětná vazba z jednotlivých schůzek

Během vývoje aplikace proběhlo několik schůzek se zástupci chirurgické kliniky z FNO, kteří nám poskytli praktické poznámky k vylepšení a případně nás více zasvětili do tématu. Zpětná vazba od lékařu byla v průběhu schůzek následující.

- Bylo nám řečeno, že volumetrické řešení je pro ně praktičtější, neboť dochází přímo k práci s DICOM daty a nedochází k tvorbě modelů.
- Lékaři by ocenili, kdyby volumetrická verze disponovala více hustotními intervaly pro dataset. Tím by bylo možné zobrazovat více intervalových oken současně.
- Užitečné by také pro lékaře bylo provádět v datasetu řezy pomocí sférického objektu.
- Segmentační modul byl v průběhu schůzek často zmiňován a schopnost zobrazit odděleně segmenty byla pro lékaře důležitá, zejména také díky tomu, že segmentační data by měla být v budoucnu díky neuronové síti snadněji dostupná.
- V budoucnu by lékaři ocenili možnost interakce více uživatelů současně (tzv. multiplayer).

Na jednu z pozdějších schůzek byly připraveny dvě verze aplikace, pro Hololens 2 a Quest 2, které bylo možné srovnat. Verze pro Quest 2 byla úspěšnější jak z hlediska ovládání, tak i v kvalitě zobrazení. Zařízení Quest 2 obsahuje display s vyšším rozlišením a větším zorným polem než Hololens 2, což vede ke kvalitnějšímu zobrazení dat. Díky VR ovladačům je ovládání také lepší a konzistentnější než u snímání rukou, které může být problematické, pokud nejsou dosaženy ideální světelné podmínky. U snímání rukou je také v některých případech detekce gesta zpožděná nebo chybná. Tyto chyby byly zejména viditelné ve školní laboratoři, kde snímání často selhávalo. Při ovládání Hololens 2 je nutné provádět gesta velmi zřetelně, aby byla rozpoznána. S tím byl obecně

problém, protože lékaři nebyli na tento způsob ovládání zvyklí. Ovládání pomocí VR ovladačů bylo pro ně intuitivnější, neboť obsahují také stisknutelná tlačítka.

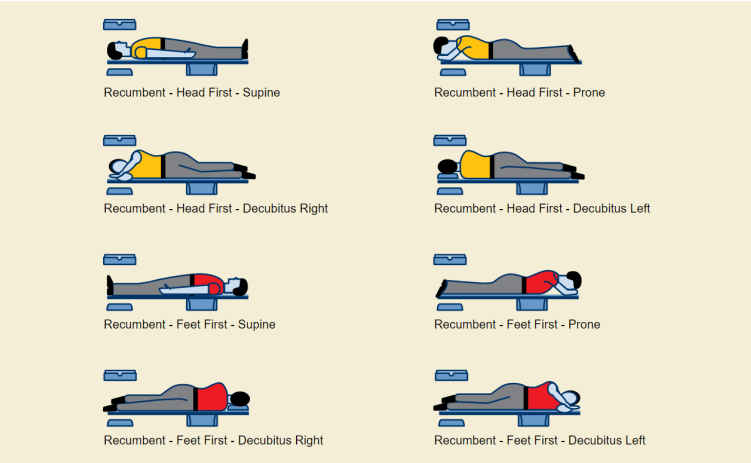
Na základě schůzek bylo rozhodnuto, že v současné době je pro lékaře verze pro Oculus Quest 2 lepší než verze pro Hololens 2 z následujících důvodů:

- Quest 2 nabízí lepší rozlišení a vyšší kvalitu zobrazení,
- Quest 2 disponuje větším zorným polem,
- cena Quest 2 je oproti Hololens 2 téměř desetinová[1],
- větší intuitivnost VR ovládání.

Na konci března roku 2023 byly dokončeny všechny zásadní plánované funkce pro aplikaci, včetně žádaného segmentačního modulu, a všechny připomínky byly patřičně zohledněny. Připravená aplikace byla společně s vypůjčeným Quest 2 a počítačem nasazena v nemocnici pro testování. Během tohoto testování si však lékaři všimli jednoho zásadního problému, který nám unikl. Problém spočíval v tom, že zobrazované datasety byly zrcadlově přetočeny, což znamenalo, že orgány, které měly být na levé straně pacienta, se nacházely na pravé.

9.1 Různé typy CT skenů

Po prozkoumání problému bylo zjištěno, že CT skeny mohou existovat v několika variantách, které jsou zobrazeny na obrázku 9.1. Vizualizační knihovna však předpokládala, že na vstup může přijít pouze dataset začínající od hlavy směrem k nohám.



Obrázek 9.1: Různé typy CT skenu (převzato z [60])

[1]Cena Quest 2 se pohybuje okolo 400-550$. Hololens 2 pak v rozmezí 3500-5200$. Cenové rozmezí se týká nových kusů a je aktuální k datu 15.4.2023

Pro správnou reprezentaci dat je nutné v aplikaci rozlišit dva hlavní typy:

- sken od nohou k hlavě (Feet first),
- sken od hlavy k nohám (Head first).

Pravděpodobně většina datasetů, pokud to není uvedeno jinak, se ve výchozím stavu skenuje od nohou k hlavě [61] (Feet first). Při procházení datasetů z volně dostupných zdrojů, ale také datasetů poskytnutých přímo z FNO, bylo zjištěno, že všechny datasety se kterými jsme mohli pracovat, jsou skenovány od nohou k hlavě. V důsledku toho měla knihovna ve výchozím stavu převážnou většinu datasetů zrcadlově přetočených (srovnání proběhlo vizuálně pomocí programu Slicer3D, odkud se braly referenční hodnoty).

Informaci o tom, jak je dataset naskenovaný, lze získat z několika DICOM parametrů. V původní verzi opravy tohoto problému aplikace využívala parametr 0020|0032 [61], který je povinný a měl by být v DICOM datasetu vždy přítomen. Při načítání datasetu se následně typ CT skenu zjistil porovnáním dvou snímků. Porovnával se první a poslední snímek datasetu a srovnávaly se hodnoty z osy **Z**. Osa **Z** v tomto konkrétním parametru uvádí podélnou pozici řezu v pacientovi. Pokud byla hodnota **Z** na posledním snímku větší než na prvním, znamenalo to, že se jednalo o dataset od nohou k hlavě. Pokud se ale hodnota **Z** postupně snižovala, jednalo se o sken od hlavy k nohám. Pokud aplikace tedy detekovala dataset od nohou k hlavě, následně celé načtené data otočila, čímž byl problém opraven a dosáhlo se správného zobrazení. Zjištění pozice snímku je vidět v kódu 9.1.

```
private bool TryGetPositionInPatient(Image sliceImage,out Vector3 position)
{
    try
    {
        List<string> metadataKeys = sliceImage.GetMetaDataKeys().ToList();

        string imagePositionPatient = sliceImage.GetMetaData("0020|0032"); //Tag slice pozice
        string [] arr = imagePositionPatient.Split('\\');

        float x = float.Parse(arr [0], CultureInfo.InvariantCulture);
        float y = float.Parse(arr [1], CultureInfo.InvariantCulture);
        float z = float.Parse(arr [2], CultureInfo.InvariantCulture);

        position = new Vector3(x, y, z);
        return true;
    }
    catch
    {
        position = Vector3.zero;
        return false;
    }
}
```

Listing 9.1: Zjištění pozice řezu v pacientovi

Tato extrakce pozic snímků a následné seřazení je potřeba pokud pracujeme s knihovnou pro čtení dat, která tuto korekci a seřazení neprovádí automaticky. Ve výchozí vykreslovací knihovně byl otevřen tento problém a při vzájemné výměně postřehů a informací jsme společně s autorem knihovny došli k závěru, že SimpleITK knihovna tyto záležitosti řeší sama automaticky podle dostupných parametrů. Interní řešení SimpleITK je však pravděpodobně velmi podobné kódu uvedeném výše, neboť obecně není mnoho možností jak daný problém správně řešit. Způsob správného seřazení byl také popsán v emailové korespondenci [62], kde Jolinda Smith (z Lewis Center for NeuroImaging University of Oregon) korektní postup vysvětluje. SimpleITK a ITK dokumentace není na toto téma moc detailní a jediná zmínka o správném seřazení kterou se mi podařilo najít je v této sekci SimpleITK dokumentace, kde je zmíněno, že dochází k vhodnému seřazení, avšak není specifikováno jak. Nedostatek informací z dokumentace je u SimpleITK a ITK bohužel docela častý, obecně i při hledání jestli knihovna provádí automatickou konverzi původních hodnot na Hounsfieldovy jednotky je informací o tématu poskromnu a musely být provedeny manuální testy, kde se automatická konverze potvrdila.

Vzhledem tedy k tomu, že knihovna SimpleITK by měla tyto datasety načíst vždy ve správném pořadí, odpadá ruční seřazení snímků a dá se předpokládat, že otočení je potřeba pro každý dataset, aby nedocházelo k zrcadlení. Pokud by se i přesto stalo, že některý dataset se načte zrcadlově, je možné dataset manuálně opravit z nastavení aplikace. Postup je vysvětlen v návodu v sekci opravy zrcadlení. Autor vizualizační knihovny pak ve své knihovně dodělal jak opravu pro SimpleITK, tak i alternativní OpenDICOM (kde snímky již manuálně řadí podle DICOM parametru 0020|0032).

Kapitola 10

Závěr

Tato diplomová práce se zaměřila na vývoj dvou odlišných aplikací, které využívají různé vizualizační metody pro zobrazení volumetrických dat, a to přímé a nepřímé vizualizace. Vývoj probíhal ve spolupráci s FNO a všechny základní funkční prvky byly pro obě aplikace úspěšně dokončeny včas. Textová část práce se pak skládá z teoretické a praktické části.

V teoretické části je proveden průzkum stávajících řešení v oboru. Čtenáři jsou seznámeni s potřebnými informacemi ohledně zobrazovacích technik, lékařských formátů, možností zarovnání a dostupných hardwarových prostředků pro AR/VR. V praktické části práce je popsán vývoj jednotlivých aplikací. Aplikace byly vyvíjeny v Unity 2020.3 s využitím MRTK knihovny pro rozšířenou realitu.

Na základě rozhodnutí lékařů byla většina prostoru věnována přímé vizualizační metodě. Tato verze aplikace je založena na kvalitní knihovně pro volumetrické vykreslování. Avšak všechny ovládací prvky knihovny musely být kompletně přepracovány pro AR/VR verzi. V aplikaci s přímou vizualizací se podařilo implementovat řadu vylepšení, mezi nejvýznamnějšími jsou: dva způsoby ovládání aplikace pomocí tzv. hand menu a jednotlivých konfiguračních panelů datasetu, asynchronní načítání a tvorba dat (pro plynulost aplikace), segmentační modul schopný jednotlivě zobrazit data k vybraným segmentům, automatický ukládací systém pro uložení důležitých nastavení u datasetu a přidání možnosti sférického výřezu pro volumetrická data. Všechny tyto vylepšení jsou popsány v jednotlivých sekcích a kapitolách praktické části této práce. Během vývoje byly některé úpravy také přidávány zpětně na GitHubu do původní vizualizační knihovny, aby je mohli využít i ostatní uživatelé ve svých projektech.

Některé vylepšení do aplikací se však již kvůli časové tísni implementovat nepodařilo. Tato časová tíseň byla způsobena velmi pozdním dodáním potřebných materiálů. Do budoucích vylepšení tedy patří: možnost interakce více uživatelů současně (tzv. multiplayer), nastavování alfa hodnot u transfer funkce ve volumetrické aplikaci, ukládací systém v modelové aplikaci a automatizace zpracování modelů v modelové aplikaci.

Obě aplikace jsou multiplatformní a dostupné pro rozšířenou i virtuální realitu. Byly specificky testovány se zařízeními Hololens 2 a Quest 2. Pro obě verze aplikace byly provedeny výkonnostní testy, které jsou popsány v praktické části textu. Volumetrická verze aplikace s VR zařízením Oculus Quest 2 byla úspěšně nasazena ve FNO koncem března 2023, kde ji lékaři od té doby testují s reálnými daty.

Celkové hodnocení lékařů je velmi pozitivní a novou možnost vizualizace CT datasetů ve VR, která umožňuje provádět různé řezy a další operace, si velmi pochvalují. Chirurgický tým FNO nám v dubnu 2023 poskytl další zpětnou vazbu se seznamem připomínek a doporučení. MUDr. Jan Roman uvedl „Většina jsou spíše drobnosti, software se mi velice líbí, vypadá to skvěle, dobře se to používá, vidím to jako plně klinicky použitelné.“.

Cílem vývoje softwaru byla jeho otevřenost a čitelnost zdrojového kódu, protože je pravděpodobné, že v budoucnu jej bude někdo přebírat a navazovat na stávající řešení. Celý zdrojový kód spolu s uživatelským a vývojářským návodem je transparentně umístěn ve veřejném repozitáři na GitHubu, což umožňuje komukoli jej použít a stavět na něm svá řešení.

Je pravděpodobné, že řešení bude dále rozvíjeno jak v souvislosti s některými současnými návrhy na vylepšení, které se již nestihly realizovat, tak i s návrhy, které přijdou v budoucnu.

Literatura

1. SILVA, Jennifer N.A.; SOUTHWORTH, Michael; RAPTIS, Constantine; SILVA, Jonathan. Emerging Applications of Virtual Reality in Cardiovascular Medicine. *JACC: Basic to Translational Science*. 2018, roč. 3, č. 3, s. 420–430. ISSN 2452-302X. Dostupné z DOI: https://doi.org/10.1016/j.jacbts.2017.11.009.
2. ECKERT, Martin; VOLMERG, Julia S; FRIEDRICH, Christoph M. Augmented Reality in Medicine: Systematic and Bibliographic Review. *JMIR Mhealth Uhealth*. 2019-04, roč. 7, č. 4, e10967. ISSN 2291-5222. Dostupné z DOI: 10.2196/10967.
3. BULLIARD, Jonas; EGGERT, Sebastian; AMPANOZI, Garyfalia; AFFOLTER, Raffael; GASCHO, Dominic; SIEBERTH, Till; THALI, Michael J.; EBERT, Lars C. Preliminary testing of an augmented reality headset as a DICOM viewer during autopsy. *Forensic Imaging*. 2020, roč. 23, s. 200417. ISSN 2666-2256. Dostupné z DOI: https://doi.org/10.1016/j.fri.2020.200417.
4. NOECKER, Angela M.; MLAKAR, Jeffrey; PETERSEN, Mikkel V.; GRISWOLD, Mark A.; MCINTYRE, Cameron C. Holographic visualization for stereotactic neurosurgery research. *Brain Stimulation*. 2023, roč. 16, č. 2, s. 411–414. ISSN 1935-861X. Dostupné z DOI: https://doi.org/10.1016/j.brs.2023.02.001.
5. BALCI, Deniz; KIRIMKER, Elvan Onur; RAPTIS, Dimitri Aristotle; GAO, Yujia; KOW, Alfred Wei Chieh. Uses of a dedicated 3D reconstruction software with augmented and mixed reality in planning and performing advanced liver surgery and living donor liver transplantation (with videos). *Hepatobiliary Pancreatic Diseases International*. 2022, roč. 21, č. 5, s. 455–461. ISSN 1499-3872. Dostupné z DOI: https://doi.org/10.1016/j.hbpd.2022.09.001. Cancer Immunotherapy in Hepatobiliary Malignancies.
6. TIGCHELAAR, Seth S.; MEDRESS, Zachary A.; QUON, Jennifer; DANG, Phuong; BARBERY, Daniela; BOBROW, Aidan; KIN, Cindy; LOUIS, Robert; DESAI, Atman. Augmented Reality Neuronavigation for En Bloc Resection of Spinal Column Lesions. *World Neurosurgery*. 2022, roč. 167, s. 102–110. ISSN 1878-8750. Dostupné z DOI: https://doi.org/10.1016/j.wneu.2022.08.143.

7. ITO, Taku; KAWASHIMA, Yoshiyuki; YAMAZAKI, Ayame; TSUTSUMI, Takeshi. Application of a virtual and mixed reality-navigation system using commercially available devices to the lateral temporal bone resection. *Annals of Medicine and Surgery*. 2021, roč. 72, s. 103063. ISSN 2049-0801. Dostupné z DOI: https://doi.org/10.1016/j.amsu.2021.103063.

8. ZOABI, Adeeb; OREN, Daniel; TEJMAN-YARDEN, Shai; REDENSKI, Idan; KABLAN, Fares; SROUJI, Samer. " Initial experience with augmented reality for treatment of an orbital floor fracture – A Technical Note ". *Annals of 3D Printed Medicine*. 2022, roč. 7, s. 100072. ISSN 2666-9641. Dostupné z DOI: https://doi.org/10.1016/j.stlm.2022.100072.

9. STEFANI, Caroline; LACY-HULBERT, Adam; SKILLMAN, Thomas. ConfocalVR: Immersive Visualization for Confocal Microscopy. *Journal of Molecular Biology*. 2018, roč. 430, č. 21, s. 4028–4035. ISSN 0022-2836. Dostupné z DOI: https://doi.org/10.1016/j.jmb.2018.06.035.

10. VELAZCO-GARCIA, Jose D.; SHAH, Dipan J.; LEISS, Ernst L.; TSEKOS, Nikolaos V. A modular and scalable computational framework for interactive immersion into imaging data with a holographic augmented reality interface. *Computer Methods and Programs in Biomedicine*. 2021, roč. 198, s. 105779. ISSN 0169-2607. Dostupné z DOI: https://doi.org/10.1016/j.cmpb.2020.105779.

11. BERNHARDT, Sylvain; NICOLAU, Stéphane A.; SOLER, Luc; DOIGNON, Christophe. The status of augmented reality in laparoscopic surgery as of 2016. *Medical Image Analysis*. 2017, roč. 37, s. 66–90. ISSN 1361-8415. Dostupné z DOI: https://doi.org/10.1016/j.media.2017.01.007.

12. LOUIS, Robert G; STEINBERG, Gary K; DUMA, Christopher; BRITZ, Gavin; MEHTA, Vivek; PACE, Jonathan; SELMAN, Warren; JEAN, Walter C. Early experience with virtual and synchronized augmented reality platform for preoperative planning and intraoperative navigation: A case series. *Operative Neurosurgery*. 2021, roč. 21, č. 4, s. 189–196. Dostupné z DOI: 10.1093/ons/opab188.

13. *Case Western Reserve, Cleveland Clinic collaborate with Microsoft on Earth-shattering mixed-reality technology for Education*. [B.r.]. Dostupné také z: https://case.edu/hololens/.

14. ECHOPIXEL, Inc. *Echopixel, Inc.* EchoPixel, Inc., [b.r.]. Dostupné také z: https://vimeo.com/user12484592?embedded=false&source=owner_name&owner=12484592.

15. TŘINEC-PODLESÍ, Nemocnice AGEL. *Kardiologové z Třince-Podlesí využívají při operacích SRDCE Rozšířenou realitu. Vyvinuli Si Vlastní software a vidí Tak Doslova „do Nitra Pacienta"*. [B.r.]. Dostupné také z: https://nemocnicetrinecpodlesi.agel.cz/o-nemocnici/novinky/220609-vr-sal.html.

16. HINUM, Klaus. *Qualcomm snapdragon 850 SOC - benchmarks and Specs*. Notebookcheck, 2019-05. Dostupné také z: https://www.notebookcheck.net/Qualcomm-Snapdragon-850-SoC-Benchmarks-and-Specs.420753.0.html.

17. HINUM, Klaus. *Qualcomm Adreno 630 GPU*. Notebookcheck, 2018-04. Dostupné také z: https://www.notebookcheck.net/Qualcomm-Adreno-630-GPU.299832.0.html.

18. VTIETO. *Use PC resources to power your app with holographic remoting remote app - mixed reality*. [B.r.]. Dostupné také z: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/use-pc-resources.

19. MURPHY, Andrew. *Hounsfield unit: Radiology reference article*. Radiopaedia.org, 2023-03. Dostupné také z: https://radiopaedia.org/articles/hounsfield-unit.

20. REEVES, TE; MAH, P; MCDAVID, WD. Deriving Hounsfield units using grey levels in cone beam CT: A clinical application. *Dentomaxillofacial Radiology*. 2012, roč. 41, č. 6, s. 500–508. Dostupné z DOI: 10.1259/dmfr/31640433.

21. INNOLITICS, LLC. *Rescale Slope Attribute*. Innolitics, [b.r.]. Dostupné také z: https://dicom.innolitics.com/ciods/digital-x-ray-image/dx-image/00281053.

22. INNOLITICS, LLC. *Rescale Intercept Attribute*. Innolitics, [b.r.]. Dostupné také z: https://dicom.innolitics.com/ciods/ct-image/ct-image/00281052.

23. MCCORMICK, Matt. *DICOM rescale intercept / rescale slope and ITK*. Kitware, 2014-10. Dostupné také z: https://www.kitware.com/dicom-rescale-intercept-rescale-slope-and-itk/.

24. PÁNKOVÁ, Ing. Olga. *Zobrazovací systémy - CT* [https://is.muni.cz/el/med/jaro2021/BRPR0422p/Zobrazovaci_systemy_-_CT.pdf]. Masaryk University, 2021. Accessed: 2023-4-3.

25. *NRRD*. [B.r.]. Dostupné také z: https://teem.sourceforge.net/nrrd/index.html.

26. *NIFTI: - neuroimaging informatics technology initiative*. U.S. Department of Health a Human Services, [b.r.]. Dostupné také z: https://nifti.nimh.nih.gov/.

27. SCHROEDER, William; MAYNARD, Robert; GEVECI, Berk. Flying Edges: A High-Performance Scalable Isocontouring Algorithm. In: 2015-10. Dostupné z DOI: 10.13140/RG.2.1.3415.9609.

28. BENTON, Alex. *GPU Ray Marching* [https://bit.ly/3GfdOlj]. University of Cambridge, Computer Laboratory, 2017-2018. Accessed: 2023-04-03.

29. TAVOOSI, Jafar; ZHANG, Chunwei; MOHAMMADZADEH, Ardashir; MOBAYEN, Saleh; MOSAVI, Amir H. Medical image interpolation using recurrent type-2 fuzzy neural network. *Frontiers in Neuroinformatics*. 2021, roč. 15. Dostupné z DOI: 10.3389/fninf.2021.667375.

30. HÄGGSTRÖM, Mikael. *Hounsfield unit (table)*. Radlines, 2019-01. Dostupné také z: https://radlines.org/Hounsfield_unit.

31. MURPHY, Andrew. *Maximum intensity projection: Radiology reference article*. Radiopaedia.org, 2023-03. Dostupné také z: https://radiopaedia.org/articles/maximum-intensity-projection.

32. UNKNOWN. *Direct Volume Rendering* [https://cgl.ethz.ch/teaching/former/scivis_07/Notes/stuff/StuttgartCourse/VIS-Modules-06-Direct_Volume_Rendering.pdf]. ETH Zurich, Computer Graphics Laboratory, 2007. Accessed: 2023-04-03.

33. TREASURE, Craig. *Azure object anchors overview - azure object anchors*. [B.r.]. Dostupné také z: https://learn.microsoft.com/en-us/azure/object-anchors/overview.

34. TREASURE, Craig. *Quickstart: Create a hololens app with unity and MRTK - Azure Object Anchors*. [B.r.]. Dostupné také z: https://learn.microsoft.com/en-us/azure/object-anchors/quickstarts/get-started-unity-hololens-mrtk?tabs=unity-package-web-ui.

35. CHEN, Kenny; GABRIEL, Paolo; ALASFOUR, Abdulwahab; GONG, Chenghao; DOYLE, Werner K; DEVINSKY, Orrin; FRIEDMAN, Daniel; DUGAN, Patricia; MELLONI, Lucia; THESEN, Thomas; AL., et. *Patient-specific pose estimation in clinical environments*. U.S. National Library of Medicine, 2018-10. Dostupné také z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6255526/.

36. TURNER, Alex. *Spatial anchors - mixed reality*. [B.r.]. Dostupné také z: https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-anchors.

37. *Working with the Hololens sample in Unity*. Vuforia, [b.r.]. Dostupné také z: https://library.vuforia.com/platform-support/working-hololens-sample-unity.

38. *Wikitude ar SDK for hololens*. 2019-08. Dostupné také z: https://www.wikitude.com/wikitude-ar-sdk-for-hololens/.

39. QIAN, Long. *HoloLens with ARToolKit*. [B.r.]. Dostupné také z: https://github.com/qian256/HoloLensARToolKit.

40. SEMPLE, Kevin. *MRTK2-Unity Developer Documentation - MRTK 2*. Microsoft, 2022-12. Dostupné také z: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05.

41. SEAN-KERAWALA. *Welcome to the mixed reality feature tool - mixed reality*. [B.r.]. Dostupné také z: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/welcome-to-mr-feature-tool.

42. SEAN-KERAWALA. *Introduction to the mixed reality toolkit–set up your project and use hand interaction - training*. [B.r.]. Dostupné také z: https://learn.microsoft.com/en-us/training/modules/learn-mrtk-tutorials/.

43. FERRONE, Harrison. *Holographic remoting overview - mixed reality*. Microsoft, 2022-09. Dostupné také z: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/native/holographic-remoting-overview.

44. KEVELEIGH. *Holographic remoting - MRTK 2*. [B.r.]. Dostupné také z: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/tools/holographic-remoting?view=mrtkunity-2022-05.

45. TREASURE, Craig. *Quickstart: Create an object anchors model to be used in an app - azure object anchors*. Microsoft, 2022-11. Dostupné také z: https://learn.microsoft.com/en-us/azure/object-anchors/quickstarts/get-started-model-conversion.

46. WEN, Qian. *QR code tracking Overview - mixed reality*. [B.r.]. Dostupné také z: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/qr-code-tracking-overview#are-qr-codes-saved-at-the-space-level-or-app-level.

47. TURNER, Alex. *Hologram stability - mixed reality*. Microsoft, 2021-10. Dostupné také z: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/hologram-stability.

48. IKITS, Milan; KNISS, Joe; LEFOHN, Aaron; HANSEN, Charles. *Chapter 39. volume rendering techniques*. Nvidia, 2007-09. Dostupné také z: https://developer.nvidia.com/gpugems/gpugems/part-vi-beyond-triangles/chapter-39-volume-rendering-techniques.

49. LAVIK, Matias; SAYS: Elektropepi. *Volume rendering in Unity*. 2020-01. Dostupné také z: https://matiaslavik.wordpress.com/2020/01/19/volume-rendering-in-unity/.

50. SOLER, L.; HOSTETTLER, A.; AGNUS, V.; CHARNOZ, A.; FASQUEL, J.; MOREAU, J.; OSSWALD, A.; BOUHADJAR, M.; MARESCAUX, J. *3D image reconstruction for comparison of algorithm database: A patient specific anatomical and medical image database*. 2010. Tech. Rep. IRCAD, Strasbourg, France. Dostupné také z: https://www.ircad.fr/research/data-sets/liver-segmentation-3d-ircadb-01/.

51. PINTER, Csaba; LASSO, Andras; SUNDERLAND, Kyle. *Segmentations: Export segmentation to labelmap volume* [https://slicer.readthedocs.io/en/latest/user_guide/modules/segmentations.html#export-segmentation-to-labelmap-volume]. 3D Slicer, 2023. Accessed: 2023-04-03.

52. INNOLITICS, LLC. *Pixel Spacing Attribute*. Innolitics, [b.r.]. Dostupné také z: https://dicom.innolitics.com/ciods/rt-dose/image-plane/00280030.

53. BARRAGAN, Noe; SHARKEY, Kent; KEVELEIGH. *Hand menu* [https://learn.microsoft.com/en-us/windows/mixed-reality/design/hand-menu]. Microsoft, 2021. Accessed: 2023-04-03.

54. WAGNER, Bill. *Asynchronous programming scenarios in C#*. Microsoft, Microsoft Learn, 2023-02. Dostupné také z: https://learn.microsoft.com/en-us/dotnet/csharp/asynchronous-programming/async-scenarios. Accessed: 2023-04-05.

55. CLEARY, Stephen. *Async/Await - Best Practices in Asynchronous Programming*. Microsoft, MSDN Magazine, 2013. Dostupné také z: https://learn.microsoft.com/en-us/archive/msdn-magazine/2013/march/async-await-best-practices-in-asynchronous-programming#async-all-the-way. Accessed: 2023-4-5.

56. *SUBTRACT (Command)*. AutoCAD, [b.r.]. Dostupné také z: https://help.autodesk.com/view/ACD/2020/ENU/?guid=GUID-14872FC1-8827-4D3B-978E-20936F9A78E5.

57. *INTERSECT (Command)*. AutoCAD, [b.r.]. Dostupné také z: https://help.autodesk.com/view/ACD/2020/ENU/?guid=GUID-904008EB-D92A-4B69-B79F-6C3A033DB3DF.

58. *Textureformat*. Unity Technologies, 2023-03. Dostupné také z: https://docs.unity3d.com/ScriptReference/TextureFormat.html.

59. VOGELSANG, Brian. *Improving VR performance using motion estimation OpenGL extensions*. Qualcomm, 2021-02. Dostupné také z: https://developer.qualcomm.com/blog/improving-vr-performance-using-motion-estimation-opengl-extensions.

60. INNOLITICS, LLC. *Patient Position Attribute*. Innolitics, [b.r.]. Dostupné také z: https://dicom.innolitics.com/ciods/ct-image/general-series/00185100.

61. INNOLITICS, LLC. *Image Position (Patient) Attribute*. Innolitics, [b.r.]. Dostupné také z: https://dicom.innolitics.com/ciods/rt-dose/image-plane/00200032.

62. SMITH, Jolinda; LORENSEN, William E. *Re: [Insight-users] DICOM example images* [https://itk.org/pipermail/insight-users/2003-September/004762.html]. 2003-09. Přístupováno dne 26. dubna 2023.

Kapitola 11

Ukázková videa

Pro všechny verze aplikací byly na závěr vytvořeny ukázkové videa zobrazující použití.

- Ukázka volumetrické aplikace - Hololens 2 (AR).
- Ukázka volumetrické aplikace - Quest 2 (VR).
- Ukázka modelové aplikace - Hololens 2 (AR).
- Ukázka modelové aplikace - Quest 2 (VR).

Kapitola 12

Přiložené soubory

V této kapitole je popsána struktura přiložených souborů pro obě aplikace. Jednotlivé přílohy se nachází v přiloženém archivu s názvem **Priloha.zip**.

12.1 Priloha 1 - volumetricka aplikace

V příloze **Priloha 1 - volumetricka aplikace** se nachází následující pod-adresáře.

- **Build** adresář, který obsahuje odkazy na již sestavené verze volumetrické aplikace.
- **Showcase** obsahuje odkazy na Youtube videa demonstrující funkčnost volumetrických aplikací pro obě platformy. Na videích je ukázáno ovládání jednotlivých aplikací z uživatelského hlediska.
- **Project** adresář obsahuje celkový rekonstruovatelný Unity projekt pro archivační účely (volumetrická verze projektu.)
- **Tutorials** Adresář v sobě obsahuje všechny zpracované návody (případně odkazy) pro volumetrickou aplikaci
- **GitHub** následně obsahuje odkaz na GitHub větev projektu.

Následující tabulka 12.1 uvádí detailní strukturu projektu s relativními cestami, kterou lze prohlédnout v GitHub repozitáři, nebo v přiloženém archivu projektu. Relativní cesty jsou brány od adresáře **Assets**. Tabulka se týká volumetrické větve aplikace.

Relativní cesta	Popis adresáře
\Images	Adresář obsahuje obrázky a sprity využité v aplikaci.
\MRTK	Adresář obsahuje MRTK komponenty, které byly upraveny přímo pro potřeby aplikace. Změny jsou vidět na těchto commitech.
\Matherials	Adresář obsahuje materiály využité v projektu, např. pro sférický výřez, pozadí ve VR atd.
\MiscAnimation	Tato složka obsahuje animace, které byly dodělány pro intervalový posuvník a pro hand menu.
\Prefabs	Složka obsahuje často využívané prefaby v aplikaci.
\QR	Adresář obsahuje celkovou logiku pro rozpoznávání QR kódu. V tomto adresáři byly některé kódy týkající se rozpoznávání upraveny. Změny jsou vidět na těchto commitech.
\Resources	Ve složce se nachází výchozí transfer funkce.
\Scripts	Adresář obsahuje všechny mnou vytvořené skripty ve volumetrické verzi aplikace. V průběhu práce nicméně došlo k určitým úpravám i externích skriptů, které zde nejsou uvedeny. Úprava těchto externích skriptů je také uvedena v popisu práce.
\VolumeRendering	V tomto adresáři se nachází volumetrická vizualizační knihovna. Na jednotlivých commitech lze vidět mnou provedené změny v průběhu vývoje.

Tabulka 12.1: Relativní cesty pro volumetrickou větev

12.2 Priloha 2 - modelova aplikace

V příloze **Priloha 2 - modelova aplikace** jsou umístěny adresáře podle stejné struktury jako ve volumetrické verzi.

Následující tabulka 12.2 uvádí detailní strukturu projektu s relativními cestami, kterou lze prohlédnout v GitHub repozitáři, nebo v přiloženém archivu projektu. Relativní cesty jsou brány od adresáře **Assets**. Tabulka se týká modelové větve aplikace.

Relativní cesta	**Popis adresáře**
\Images	Adresář obsahuje obrázky a sprity využité v aplikaci.
\Matherials	Adresář obsahuje materiály využité v projektu, např. pro řezací rovinu, pozadí ve VR atd.
\OpenFracture-main	Adresář obsahuje OpenFracture knihovnu, která byla modifikována pro potřeby aplikace. Modifikace jsou vidět na těchto commitech.
\Prefabs	Složka obsahuje často využívané prefaby v aplikaci.
\QR	Adresář obsahuje celkovou logiku pro rozpoznávání QR kódu. V tomto adresáři byly některé kódy týkající se rozpoznávání upraveny. Změny jsou vidět na těchto commitech.
\Resources	Tato složka slouží pro vložení FBX modelů, které jsou zobrazeny v aplikaci.
\Scripts	adresář obsahuje všechny mnou vytvořené skripty v modelové verzi aplikace. V průběhu práce nicméně došlo k určitým úpravám i externích skriptů, které zde nejsou uvedeny.

Tabulka 12.2: Relativní cesty pro modelovou větev