ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Porazil** Jméno: **Vít** Osobní číslo: **510634**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra teorie obvodů**

Studijní program: **Lékařská elektronika a bioinformatika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Nástroj pro značení elektrod stereoencefalografie v tomografických snímcích hlavy

Název bakalářské práce anglicky:

Toolbox for stereoenecephalography electrodes labelling in head tomography

Pokyny pro vypracování:

Standardním vyšetřením chirurgické léčby epilepsie je explorace podezřelých mozkových struktur pomocí nitrolebních stereoencefalografických elektrod (SEEG). Pozice snímacích kontaktů ve vztahu k anatomickým strukturám je u každého pacienta individuální, nicméně jejich znalost je nezbytná pro správnou interpretaci EEG signálů z nich. Vzhledem ke vzrůstající složitosti onemocnění a rostoucímu počtu implantovaných elektrod je nutné identifikaci pozic elektrod automatizovat. Modifikujte již vyvinutý algoritmus [1] využívající zpracování snímků CT a MRI a implementujte jej jako rozšíření (extension) pro program 3D Slicer [2].
1) Nastudujte si dokumentaci pro tvorbu rozšíření programu 3D Slicer
2) Vytvořte uživatelsky přívětivé grafické rozhraní a implementuje potřebné algoritmy zpracování obrazu
3) Registrujte snímky CT a MRI, segmentujte oblast zájmu nitrolebního prostoru v MRI a identifikujte kovové elektrody v CT
4) Dle značky na konci elektrody trasujte elektrodu a označte jednotlivé kontakty
5) Funkčnost algoritmu otestujte na kohortě pacientů

Seznam doporučené literatury:

[1] Janca R, Tomasek M, Kalina A, Marusic P, Krsek P, Lesko R. Automated Identification of Stereoelectroencephalography Contacts and Measurement of Factors Influencing Accuracy of Frame Stereotaxy. IEEE J Biomed Heal Informatics. 2023;27(7):3326–36. https://doi.org/10.1109/JBHI.2023.3271857.
[2] Narizzano M, Arnulfo G, Ricci S, Toselli B, Tisdall M, Canessa A, et al. SEEG assistant: A 3DSlicer extension to support epilepsy surgery. BMC Bioinformatics. 2017;18(1):1–13. https://doi.org/10.1186/S12859-017-1545-8/FIGURES/6.
[3] Lucas, A., Scheid, B. H., Pattnaik, A. R., Gallagher, R., Mojena, M., Tranquille, A., ... & Sinha, N. (2023). iEEG-recon: A Fast and Scalable Pipeline for Accurate Reconstruction of Intracranial Electrodes and Implantable Devices. medRxiv, 2023-06. https://doi.org/10.1101/2023.06.12.23291286

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Radek Janča, Ph.D. katedra teorie obvodů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **06.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

Ing. Radek Janča, Ph.D. podpis vedoucí(ho) práce	doc. Ing. Radoslav Bortel, Ph.D. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta

F3

Fakulta elektrotechnická
Katedra teorie obvodů

Bakalářská práce

Nástroj pro značení elektrod stereoencefalografie v tomografických snímcích hlavy

Vít Porazil
Lékařská elektronika a bioinformatika

Květen 2024
Vedoucí práce: Ing. Radek Janča, Ph.D.

Poděkování / Prohlášení

Chtěl bych poděkovat svému vedoucímu za jeho trpělivý přístup a mé rodině, za nikdy nekončící podporu. Také bych chtěl poděkovat pracovníkům Centra pro epilepsie Fakultní Nemocnice v Motole za poskytnutá data.

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

V Praze dne

.....

Abstrakt / Abstract

Monitorace mozkové aktivity pomocí nitrolebních stereoencefalografických elektrod (SEEG) je jedním ze standradních vyšetření chirurgické léčby epilepsie. Znalost pozic stovek snímacích elektrod je nezbytná pro správnou interpretaci EEG signálu, avšak jejich získání je z obrazů výpočetní tomografie bez automatického zpracování obtížné. V rámci této práce byl modifikován dříve publikovaný detekční algoritmus, který byl následně implementován v podobě rozšíření pro program 3D Slicer. Tento nástroj pro automatickou detekci kontaktů SEEG elektrod byl otestován na skupině šesti pacientů. Výsledek testování ukazuje, že nástroj je schopný detekovat kontakty s přesností 0.21 ± 0.10 mm.

Klíčová slova: SEEG detekce, výpočetní tomografie , 3D Slicer, zpracování obrazu

Monitoring of brain activity with intracranial stereoencephalographic electrodes (SEEG) is one of the standard tests in the surgical treatment of epilepsy. Knowledge of the positions of hundreds of scanning electrodes is essential for correct interpretation of the EEG signal, but their extraction from computed tomography images is difficult without automatic processing. In this work, a previously published detection algorithm was modified and subsequently implemented as an extension for the 3D Slicer program. This tool for automatic detection of SEEG electrode contacts was tested on a group of six patients. Results of testing shows that the tool is able to detect contacts with accuracy of 0.21 ± 0.10 mm.

.

Keywords: SEEG detection, computed tomography, 3D Slicer, image processing

Title translation: Toolbox for stereoenecephalography electrodes labelling in head tomography

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

1.1 Představení problému

Epilepsie, jakožto neurologické onemocnění, zasahuje naší populaci s přibližnou prevalencí kolem jednoho procenta [1]. V případech fokální epilepsie rezistentní na farmakoterapii je chirurgický zákrok často zvažován jako terapeutická alternativa. Zejména resekční chirurgie je stále častěji preferovaným přístupem, nabízejícím až osmdesátiprocentní míru úspěšnosti. [2].
Aby se zaručil úspěch operace, je třeba co nejpřesněji vymezit hranice epileptogenní oblasti. V některých případech lze zájmovou oblast určit pomocí neivazivních metod (EEG, snímky magnetické rezonance), nicméně v řadě případů tyto metody nedokáží přesně určit zájmovou oblast, nebo si jejich výsledky navzájem odporují. V takové situaci je třeba se uchýlit k metodám invazivním [3]. Mezi nimi hraje klíčovou roli intrakraniální elektroecefalografie, konkrétně stereo-encefalografie (SEEG), která je preferovanou variantou pro svou bezpečnost a efektivitu [4]. Při SEEG dochází k zavedení nitrolebních elektrod do mozku pacienta pro snímání lokálních potenciálů, čímž je umožněno epileptogenní oblasti mapovat s vysokým prostorovým rozlišením.
Pro správnou interpretaci záznamu SEEG monitorice je nezbytná znalost vztahu snímací elektrody a anatomické struktury mozku. Tu je možno získat z postoperačního neurozobrazení mozku kombinující snímek hlavy s metalickými elektrodami pomocí výpočetní tomografie (CT) registrované se strukturálním zobrazením mozku pomocí magnetické rezonance (MRI). S pokrokem v medicíně a s nárůstem počtu implantovaných elektrod s až stovkami snímacích kontaktů je již extrémně obtížné identifikovat přesné pozice nahrávacích kontaktů manuálně, neboť manuální identifikace kontaků vyžaduje zkušenosti s prací s trojdimenzionálními zobrazovacími programy a je náchylné k chybám. [5]
Bylo učiněno několik pokusů k vytvoření semiautomatických nástrojů určených k plnění tohoto úkolu [5–7], ale žádný z nich nebyl dostatečně spolehlivý a nedosahoval uskopojivých výsledků. Potřeba pro vytvoření nástroje, který tento úkol zvládne efektivně a spolehlivě vyřešit tedy do jisté míry přetrvává. V roce 2023 byl autory Janča a kol. [8] navržen algoritmus, který vykazuje vysokou spolehlivost detekce
V této práci se pokusím na snažení předchozích výzkumů navázat, a za využití výhod, které přináší platforma 3D Slicer přivést k existenci aplikaci, která danou úlohu zvládne plnit konzistentě a s minimálním zatížením pro uživatele.
Páteř aplikace bude tvořit výše zmíněný algoritmus [8]. Tento algoritmus nabízí velmi robustní řešení pro identifikaci kontaktů elektrod a to i v případech, kde jiné algoritmy selhávali (například při křížení elektrod nebo při přítomnost artefaktů způsobených rentgenem), ale chybí mu přívětivé uživatelské rozhraní.

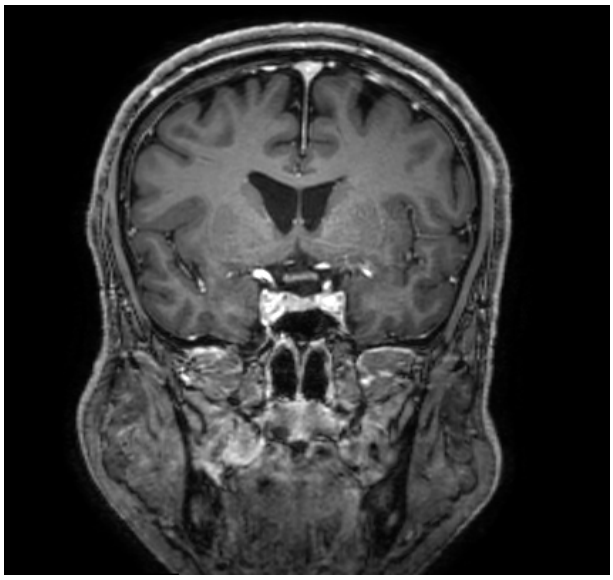
1.2 Motivace a cíl

Tato práce si dává za cíl importovat výše zmíněný algoritmus z programovacího jazyka Matlab, ve kterém byl původně vytvořen, do jazyka Python, který je s platformou Slicer kompatibilní a využít této platformy pro zpřístupnění algoritmu širší odborné veřejnosti. Pokud se tento cíl podaří splnit, vytvořený software má potenciál najít uplatnění nejenom v klinické praxi chirurgické léčby epilepsie, ale také v oblastech kognitivního výzkumu, které také využívají intrakraniální záznamy EEG a vyžadují přesnou znalost pozice elektrod.

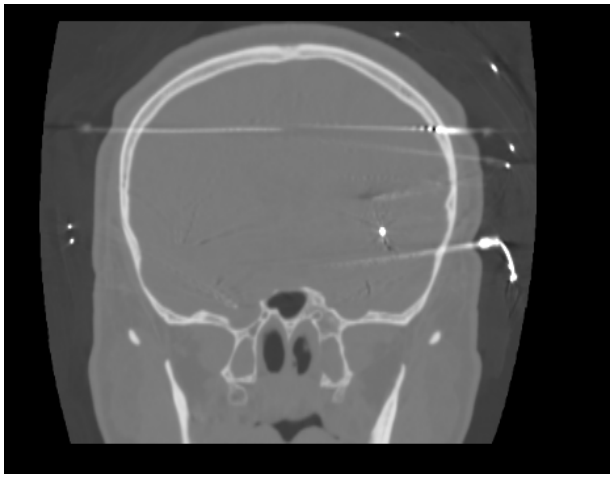
Kapitola 2
Návrh aplikace

2.1 Požadavky na aplikaci

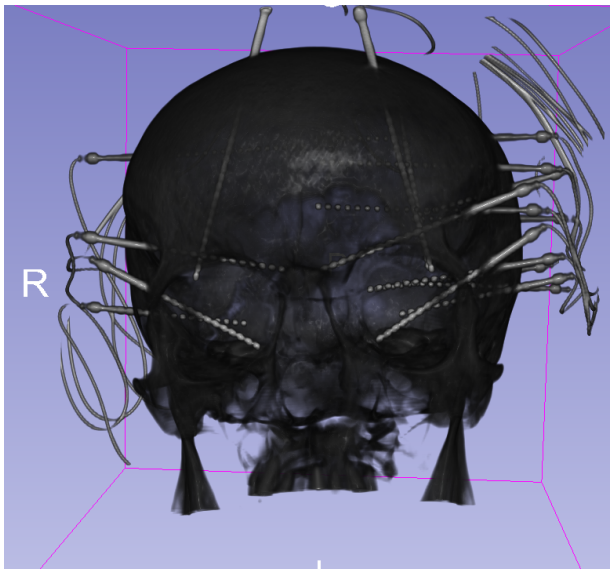
Aplikace, která v rámci této práce vznikne musí splnit tři hlavní požadavky.
Prvním z požadavků, je úspěšná implementace detekčního algoritmu. Ten by měl operovat následovně. Aplikace na vstupu dostane dva trojrozměrné snímky mozku a hlavy. Prvním snímkem je pre-implantační magnetická rezonance (MRI), sloužící jako referenční snímek mozku definující zájmovou oblast, ve které budou elektrody vyhledávány. Druhým snímkem je post-implantační výpočetní tomografie (CT). Na tomto snímku je zachycena celá pacientova hlava, včetně implantovaných elektrod a přívodních drátů, které k elektrodám vedou, a kotvících šroubů, které fixují elektrody v lebeční kosti.
Dále aplikace dostane na vstupu seznam základních informací o elektrodách: bodů specifikujících pozice vstupu jednotlivých elektrod do lebky (tj. pozice šroubů), spolu s informací o délkách elektrod, respektive počtu kontaktů na elektrodě. Uživatel by měl mít možnost tento seznam jednoduše vytvořit ručně přímo v rámci aplikace.
Aplikace tyto vstupy zpracuje a vrátí seznam bodů, ve kterém budou obsaženy koordináty všech kontaktů na všech elektrodách. Uživatel nakonec musí mít možnost tento seznam uložit pro pozdější využití.
Druhým z požadavků, je implementace přívětivého uživatelského rozhraní přizpůsobeného k plnění představeného úkolu.
Třetím z hlavních požadavků je požadavek na dostupnost. Aplikace by měla být volně a snadno dostupná každému, kdo má zájem o její využití a zároveň by k jejímu použití neměl být vyžadován žádný další placený, či obskurní software.



Obrázek 2.1. Ukázka snímku magnetické rezonance.



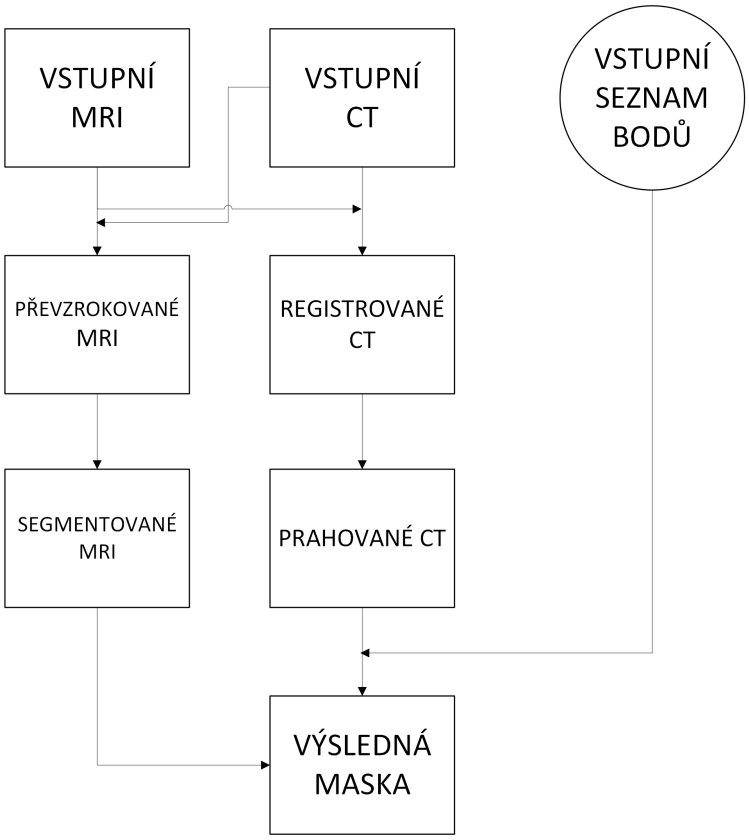
Obrázek 2.2. Ukázka snímku výpočetní tomografie.



Obrázek 2.3. Elektrody a šrouby po implantaci.

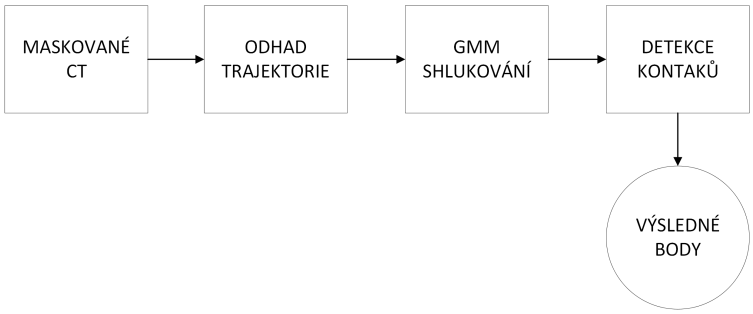
2.2 Představení algoritmu

Naplnění prvního z požadavků aplikace dosáhne modifikací již dříve zmíněného algoritmu [8], jehož schéma je uvedeno na obrázcích níže. Pro přehlednost jsem schéma rozdělil na dvě části a to na první část, ve které jsou prováděny operace, které vedou k izolaci elektrod z původního snímku a druhou část, ve které dochází k matematické aproximaci elektrod a následné identifikaci jednotlivých kontaktů. Schéma se zde pokusím alespoň přibližně vysvětlit, podrobnosti o každé části jsou uvedeny v kapitole implementace.



Obrázek 2.4. Schéma první části algoritmu.

V první části algoritmu jsou snímky MRI a CT upraveny tak, aby s nimi bylo co nejjednodušší pracovat. Nejprve jsou registrovány, následně je jeden ze snímků přeškálován, tak aby oba snímky měli stejné rozlišení.
Jelikož CT obsahuje hodně artefaktů a elektrody mají konektory a kabely, které detekci elektrod komplikují, je využita segmentace MRI, pro omezení oblasti zájmu. Do oblasti zájmu také patří kotvící šrouby, které jsou určeny pomocí vstupních bodů specifikovaných uživatelem. Výsledkem první části algoritmu je prahované a maskované CT obsahující pouze kovové prvky nacházející se v oblasti zájmu.



Obrázek 2.5. Schéma druhé části algoritmu.

V druhé části jsou extrahovány jednotlivé elektrody a jejich kontaky označeny. Pomocí kotvících šroubů elektrod lze odhadnout směr implantovaných elektrod a ze znalosti délky elektrod lze odhadnout jejich přibližnou trajektorii. Trajektorie jsou využity pro inicializaci Gaussova směsového modelu, pomocí kterého se jednotlivé elektrody

označí. Poté může být každá elektroda zpracována samostatně. Každá elektroda je proložena polynomem, na kterém jsou následně hledány jednotlivé kontakty. Algoritmus nakonec vrátí seznam bodů specifikující pozice jednotlivých kontaktů.

2.3 Platfroma 3D Slicer

Platfroma 3D Slicer je v této práci využita, neboť nabízí skvělou cestu pro splnění požadavků na přívětivé uživatelské rozhraní a dostupnost aplikace.
3D Slicer je volně dostupný, open-source software užívaný pro vizualizaci, segmentaci, registraci a analýzu lékařských, biomedicínských a jiných vícerozměrných obrazů a pro plánování a navigaci při zákrocích řízených obrazem [9].
Tento software byl původně vyvinut Laboratoří chirugického plánování, *Brigham and Women's hospital*. Později se do projektu zapojilo mnoho dalších institucí a nyní je široce užíván v komunitě lidí zabývajících se lékařským zobrazováním [10].
3D Slicer se používá prostřednictvím desktopové aplikace, která kromě již výše zmíněných funkcionalit nabízí více než sto padesát dalších rozšíření, která si uživatel může doinstalovat podle potřeby. Pro tuto práci je ovšem nejzásadnější, že 3D Slicer zároveň nabízí možnost vytvořit vlastní rozšíření, které lze následně rovnou v rámci 3D Sliceru používat.

2.4 Vlastní rozšíření pro 3D Slicer

Doinstalovatelná rozšíření pro 3DSlicer typicky obsahují jeden, nebo více modulů, které Slicer rozšíří o další funkcionalitu. Slicer podporuje několik typů modulů, zde je krátce představím.
Prvním z podporovaných modulů jsou moduly typu *Command Line Interface (CLI)*, neboli moduly rozhraní příkazového řádku. Moduly tohoto typu jsou přístupné pouze skrze příkazový řádek a nenabízí žádné uživatelské rozhraní. Jsou používány převážně na implementaci komputačně náročných algoritmů, neboť je lze implementovat v libovolném jazyce a se Slicerem komunikují pouze skrze vstupy a výstupy. Nevýhodou je, že nejsou schopny vracet mezivýsledky.
Druhým typem podporovaných modulů jsou moduly typu *Loadable*. Jedná se o pluginy psané v jazyce C++, které se používají pro implementaci komplexních interaktivních komponentů, u kterých je kladen důraz na jejich výkonnost.
Posledním typem podporavaných modulů jsou moduly *Scriptable*. Tyto moduly jsou scripty v jazyce Python, které plně využívají Slicer API (=application programming interface). Používají se pro prototypování a na vývoj vlastních pracovních postupů. Mají plný přístup ke grafickým knihovnám MRML, VTK, Qt a ITK, tudíž se dají snadno použít k implementaci grafického rozhraní, navrženého přímo na míru pro daný úkol [11]. Mé rozšíření bude implementovat přesně tento typ modulu, neboť nejlépe naplňuje mé požadavky.

2.5 Výběr programovací jazyka

Jak již bylo zmíněno výše, mé rozšíření pro Slicer implementuje modul typu Scripted, který je vždy psán v jazyce Python. Výběr programovacího jazyka je tak učiněn za mně.

Jazyk Python byl vytvořen nizozemským programátorem Guidem van Rossumem v roce 1991. Jedná se o open-source vysokoúrovňový jazyk, který je znám především pro svou jednoduchost a snadnou čitelnost. Mezi jeho hlavní síly patří interpretabilita, využití dynamického typování a jeho versitilita. Dá se použít na řadu aplikací jako jsou například webové aplikace, desktopové aplikace, strojového učení a umělé inteligence nebo, jako je tomu z části i v mém případě, k vědeckým a numerickým výpočtům. Jedním z důvodů této široké použitelnosti je skutečnost, že Python podporuje různé programovací paradigma, umožňující vývojářům volbu mezi objektově orientovaným, imperativním a funkcionálním programováním v závislosti na potřebách konkrétní aplikace [12].

Další významnou výhodou jazyka Python je jeho rozsáhlá komunita. S nárůstem popularity Pythonu jako preferovaného programovacího jazyka se rozrůstá i komunita programátorů, kteří poskytují ucelenou dokumentaci, výukové materiály a zejména velké množství dostupných knihoven[13]. Z tohoto faktu budu v mé práci těžit i já a to hlavně díky použití matematických knihoven *Numpy* a *scikit-learn* [14] a pak také třeba knihoven k tvoření uživatelského rozhraní jako je *PyQt* nebo knihovny *vtk* k zobrazování 3D renderů a vizualizací.

Na druhou stranu je nutné uvést i některé nevýhody spojené s použitím jazyka Python. Nejvýraznější nevýhodou je jeho nižší výkonnost ve srovnání s jinými programovacími jazyky, jako jsou C a C++ [12]. To může mít za následek pomalejší provádění komputací, což může mít negativní vliv na uživatele používající aplikace na Pythonu postavené.

Nicméně výhody použití Pythonu pro tuto aplikaci zdaleka převyšují jeho nevýhody a proto bych pro její vývoj jiný jazyk nezvolil, i kdybych takovou možnost měl.

Kapitola 3
Implementace

3.1 Programování v kontextu 3D Sliceru

Předtím než zde vysvětlím jak se postupovalo při implementaci jednotlivých částí algoritmu je nutné nejprve přiblížit přiblížit jakým způsobem se pracuje s daty v rámci platformy 3D Slicer a jak komunikuje implementované rozšíření s celou aplikací.
Data jsou ve Sliceru strukturována podle datové struktury MRLM (Medical Reality Markup Language). Na vrcholu stojí takzvaná MRML scéna, která v sobě obsahuje veškeré další datové prvky a řídí vztahy mezi nimi.
Stavební kameny pak tvoří MRML uzly (nodes). Uzlů je několik typů, každý z nich specifikuje jiný typ dat. Mezi nejčastěji používané patří `vtkMRMLScalarVolumeNode`, který obsahuje informace o snímku, `vtkMRMLTransformNode` obsahující transformace a `vtkMRMLMarkupsFiducialNode` reprezentující kontrolní body.
Veškerá data, která budou textu této kapitoly zmíněna jsou ve formě některého z typu uzlů, pokud není explicitně zmíněno, že tomu je jinak.
Rozšíření s aplikací Slicer komunikuje právě skrze tyto uzly a to tak, že si některý z uzlů převezme, upraví data, která uzel obsahuje a nový uzel předá zpět do MRML scény.
Alternativně může rozšíření vytvořit úplně nový uzel v rámci MRML scény.
Rozšíření také může specifikovat jestli a jak se mají uzly vykreslovat v rámci uživatelského rozhraní. O proces zobrazení se pak stará Slicer sám, tudíž tuto část není nutno při implementaci uvažovat.

3.1.1 Souřadnicové systémy

Slicer defaultně pracuje s RAS (Right, Anterior, Superior) souřadnicovým systémem. To je potřeba vzít v potaz při implementaci funkcí, které extrahují objemová data z MRLM uzlu a s tímto tenzorem přímo pracují. V takovém případě je potřeba znát transformaci mezi IJK souřadnicovým systémem, který bude v takovém případě používaný na práci s daty a anatomickým souřadnicovým systémem.
Transformace z vektorového prostoru obrazu $(ijk)'$ do anatomického vektorového prostoru je afinní transformace složená z lineární transformace A a translace $\vec{t}$

$$\vec{x} = A(ijk)' + \vec{t} \tag{1}$$

Transformační matice A je 3×3 matice, která obsahuje informaci o rotaci prostoru a škálování os.
$\vec{t}$ je 3×1 vektor obsahující informaci o geometrické pozici prvního voxelu, tedy translaci. Matematickou úpravou lze informaci o translaci zahrnou do matice A, pokud dojde k jejímu rozšíření řádek nul a sloupec obsahující translaci zakončený jedničkou. Vektory $\vec{x}$ a ijk taky musí být rozšířeny o řádek obsahující jedničku. Matice transformace bude

tedy vypadat takto [15].

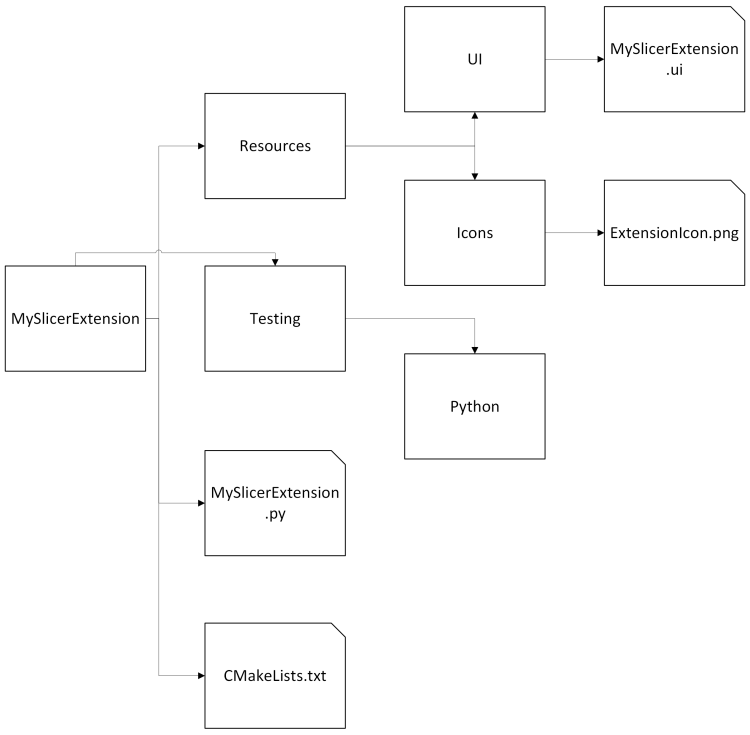
$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & t_1 \\ A_{21} & A_{22} & A_{23} & t_2 \\ A_{31} & A_{32} & A_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} i \\ j \\ k \\ 1 \end{pmatrix} \tag{2}$$

Slicer nabízí funkcionalitu pro získání této matice.
Při práci v IJK souřadnicích je důležité si uvědomit, že jeden voxel neodpovídá jednomu milimetru a při implementování fukcionality pracující s konkretní vzdáleností určenou v milimetrech je potřeba tuto vzdálenost převést na počet voxelů.

3.2 Struktura adresáře

Aplikaci je vyvíjena jakožto rozšíření pro 3D Slicer, struktura adresáře tedy bude odpovídat konvenci, která se v rámci platformy používá [16]. Celou strukturu zastřešuje složka, která typicky sdílí jméno s rozšířením (ve schématu znázorněna jako MySlicerExtension). Uvnitř této složky jsou následující objekty.

- Soubor MySlicerExtension.py obsahující zdrojový kód rozšíření.
- Konfigurační soubor CMakeList.txt, specifikující požadavky pro kompilaci projektu platformou CMake [17].
- Složka Resources, která obsahuje dvě podsložky. První z nich je podložka UI ve které se nachází soubor MySlicerExtension.ui definující grafické rozhraní rozšíření. Druhá je pak podsložka Icons, ve které se nachází ikona rozšíření, viditelná v prostředí Slicer.
- Složka Testing, ve které se nachází podsložka Python, typicky obsahující testovací skripty, případně data k provedení testů potřebná.

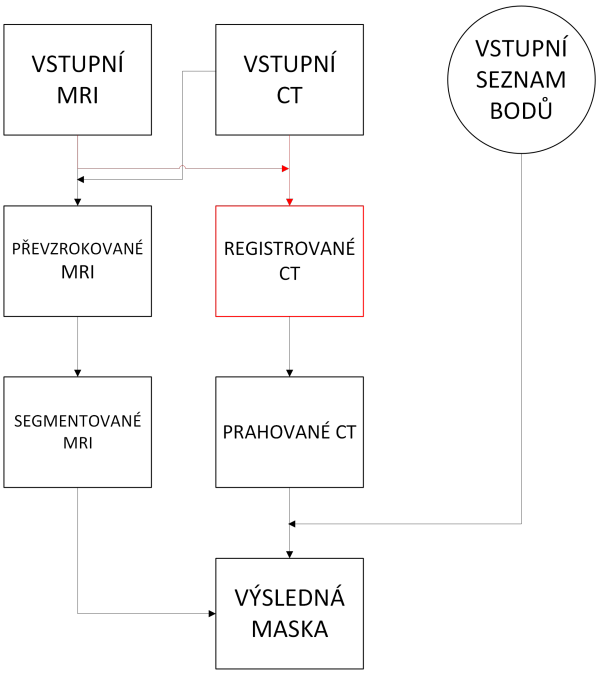


Obrázek 3.1. Schéma reprezentující strukturu adresáře projektu. Obdélníky představují složky, obdélníky s chybějícím rohem představují konkrétní soubory.

3.3 Implementace algoritmu - první část

Následující operace mají za cíl upravit snímek CT tak aby obsahoval pouze části elektrod nacházející se uvnitř mozku pacienta a šrouby, kterými elektrody do mozku vstupují.

3.3.1 Registrace



Obrázek 3.2. Operace registrace v kontextu první části algoritmu.

Vzhledem k tomu, že vstupní snímky MRI a CT jsou pořízeny na dvou různých zařízeních, je možné, že pacientova hlava je v každém z nich zachycena v jiné pozici. To samozřejmě není pro průběh algoritmu vůbec žádoucí, neboť chceme jeden snímek používat jako referenci pro práci s druhým snímkem, což jednoduše není možné, pokud se snímky nepřekrývají. Z tohoto důvodu je nutné v prvním kroku algoritmu snímky registrovat.
Registrace je proces, který má za cíl najít takovou transformaci, aby po její aplikace na snímek tento snímek co podle určitého kritéria podobnosti nejvíce odpovídal druhému snímku, který byl určen jako reference pro registraci. Existuje několik kritérií, které lze zvolit. Jednou z možností je kritérium uvažující sumu absolutních rozdílu (SAD). Matematicky lze toto kritérium vyjádřit takto.

$$\sum_x \sum_y |A(x,y) - B(x,y)| \tag{3}$$

Kde A a B jsou matice reprezentující obrazy a x a y jsou souřadnice použité na indexování obrazů. Sumy vyjadřují sčítání přes všechny tyto souřadnice. Tato rovnice problém specifikuje pro dvourozměrné obrazy, nicméně třetí rozměr doplníme analogicky.
Druhým z kriterií, které zde uvedu, je kritérium maximalizující sumu umocněných rozdílů (SSD)

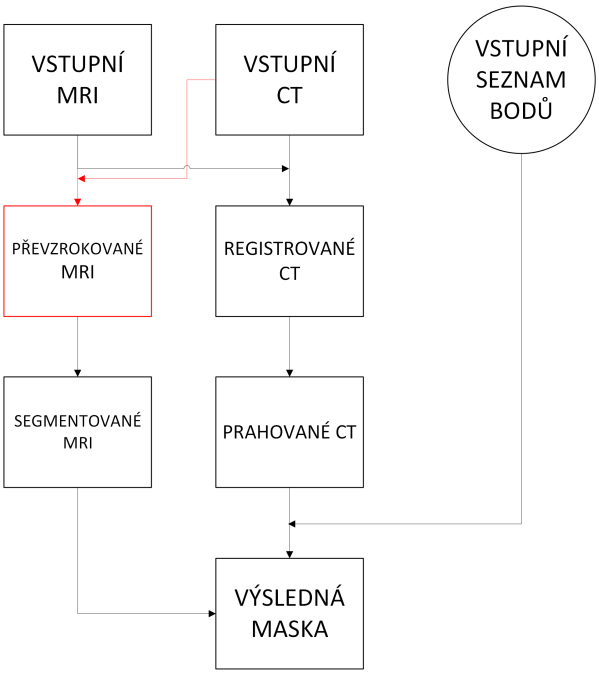
$$\sum_x \sum_y (A(x,y) - B(x,y))^2 \tag{4}$$

Značení odpovídá značení z předchozí rovnice.
Kromě výběru kritéria registrace existuje ještě mnoho dalších faktorů, které je potřeba při implementaci registrace zahrnout a implementace se tím stává poněkud náročnou. Naštěstí je ve 3D Sliceru dostupný modul BRAINSFit [18], díky kterému jsem se této strasti vyhnul.
Při volání modulu je třeba specifikovat širokou řadu parametrů. Ty nejzásádnější z nich zde uvedu a vysvětlím, jaké hodnoty jsem zvolil a proč.

- fixedVolume: specifikuje který ze snímků bude referenční, volím MRI snímek
- movingVolume: specifikuje pro který ze snímků se hledá transformace, volím CT snímek
- samplingPercentage: specifikuje jak velké procento voxelů bude použito při hledání transformace. Já jsem zvolil 1 % . To je dostatečné pro nalezení vhodné transformace, ale zároveň dostatečně malé, aby komputace nebyla příliš výkonnostně náročná.
- useRigid: určuje, zdali se použije rigidní transformace, tedy transformace uvažující pouze translaci a rotaci. Tento parametr nastavuji na hodnotu True
- initializeTransformMode: určuje, jakým způsobem bude inicializováno výchozí postavení snímku, volím hodnotu `useMomentAlign`, čímž se zajistí inicializace v težištích snímků
- outputTransform: specifikuje, kam se uloží výstupní transformace

Zbytek parametrů nechávám v defaultním nastavení. Výsledná transformace je uložena do mezipaměti programu a načtena jako uzel do Slicer prostředí, abychom k ní měli přístup v dalších krocích algoritmu.

3.3.2 Převzorkování

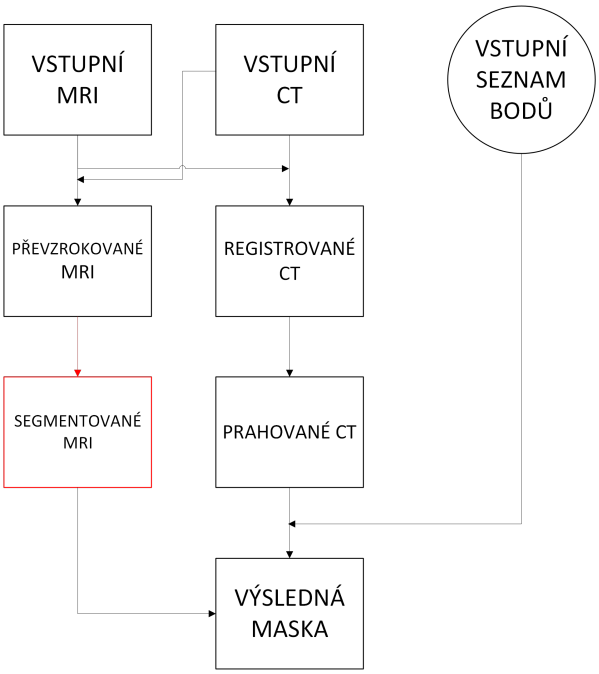


Obrázek 3.3. Operace převzorkování v kontextu první části algoritmu.

Dalším z kroků je přeškálování MRI snímku tak, aby odpovídal CT snímku. Aby bylo možné oba snímky indexovat stejnými souřadnicemi, je nutné aby snímky měli stejné rozlišení a společný počátek. Tento krok velmi usnadní aplikaci masky v jednom

z následujících kroků.
Implementace tohoto algoritmu je opět vynechána, neboť Slicer k tomuto účelu nabízí modul Resample, který jsem využil. Konkrétně jsem použil metodu *ResampleVolumeToReferenceVolume*, do které stačí jako vstupy vložit MRI a CT snímky. Metoda transformaci na MRI snímek rovnou aplikuje. Tím je tento krok vyřešen.

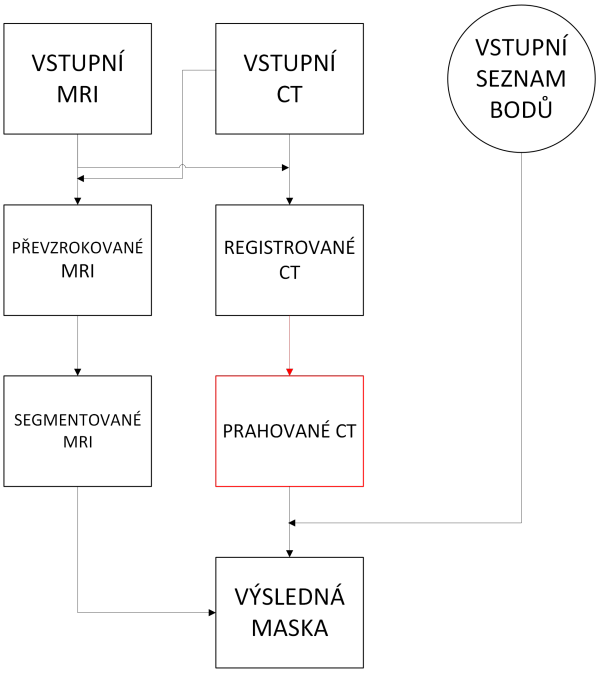
3.3.3 Segmentace



Obrázek 3.4. Operace segmentace v kontextu první části algoritmu.

Segmentací se v kontextu lékařské analýzy obrazu myslí rozdělení obrazu na jednotlivé segmenty reprezentující konkrétní anatomické struktury. Problém segmentace je jedním z nejnáročnějších v lékařském prostředí. K problému implementace lze přistoupit mnoho způsoby. Implementačně nejjednodušší algoritmy pracují pouze na základě intenzit (prahování, seskupování oblastí). Složitější řešení přistupují k problému s využitím metod strojového učení, nebo čím dál tím častěji s využitím hlubokého učení. Velmi efektivní jsou i hybridní metody, které kombinují více přístupů dohromady [19]. V mé aplikaci se jako nejvhodnější řešení ukázalo opět využít funkcionalitu Sliceru, tentokrát v podobě doinstalovatelného rozšíření HDBrainExtractionTool. Toto rozšíření vyžívá pokročilé metody založené na technikách strojového učení pro identifikaci a extrakci mozkových struktur [20].
Já rozšíření používám pro odstranění lebky a všech dalších nežádoucích struktur z MRI snímku tak, aby se zachoval pouze mozek. Použití je opět velmi snadné, stačí specifikovat vstup a výstup, o vše ostatní se nástroj postará sám. Nově vytvořený snímek, který obsahuje pouze mozek si uložím pro pozdější použití, v rámci vytváření masky pro snímek CT.

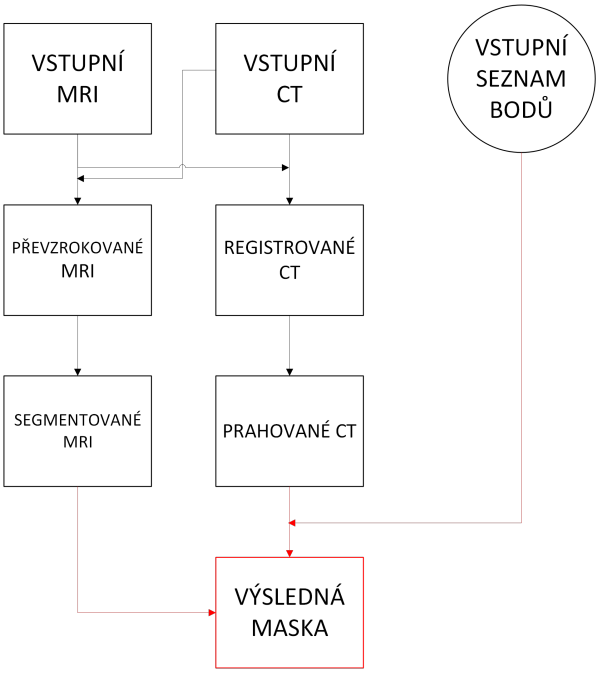
3.3.4 Prahování



Obrázek 3.5. Operace prahování v kontextu první části algoritmu.

Prahování je samo o sobě velmi jednoduchý proces. Provádí se tak, že se každému pixelu, respektive voxelu obrazu přiřadí hodnota buď jedna nebo nula, na základě toho, zda-li je jeho intenzita nad, nebo pod předem specifikovaným prahem. Nevýhodou tohoto procesu je, že se při něm obrázek převede na binární a tudíž se ztratí informace o zachovaných intenzitách. Z tohoto důvodu je vhodné prahovaný obrázek nepoužívat jako výstup, ale pouze jako masku a přemaskovat s ním obrázek původní. Tímto se zaručí to, že se veškeré voxely s intenzitou pod prahem vynulují a voxely s intenzitou nad prahem se zachovají nezměněny.
Tento proces je implementačně nenáročný, nicméně i tak jsem se rozhodl opět využít funkcionalitu Sliceru a využít modulu Threshold Scalar Volume. Jediné co tak musím řešit je hodnota prahu, nicméně vzhledem k tomu, že cílem prahování je ze snímku CT odstranit vše kromě kovových elektrod stačí hodnotu prahu nastavit tak, aby byla ze snímku odprahována veškerá tkáň. Toho docílím tak, že nastavím práh na hodnotu 3000 HU [21].

3.3.5 Vytvoření masky



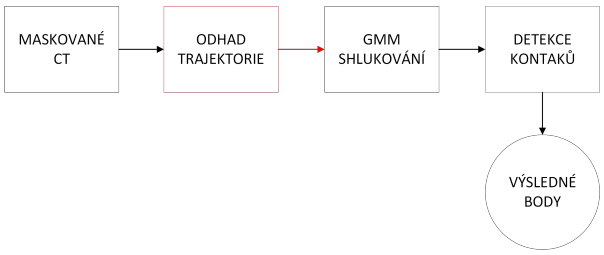
Obrázek 3.6. Operace vytvoření masky v kontextu první části algoritmu.

Maska, po jejíž aplikaci na snímek CT získáme požadovanou oblast zájmu se skládá ze dvou částí.
První část masky je oblast mozku. Vzhledem k tomu, že se snažíme identifikovat epileptogenní oblast tak nás nezajímají jiné části elektrod, než ty, které se v oblasti mozku přímo nacházejí. Pro získaní této části masky již máme předpřipravený segmentovaný snímek MRI, který stačí převést do binární formy a můžeme ho díky všem předešlým operacím (registrace, přeškálování) rovnou použít jako masku.
Aby bylo možné provést správnou aproximaci elektrod, je potřeba v masce zahrnou i jejich kotvící šrouby. U těchto šroubů je pozice přesně známá (je specifikovaná uživatelem před zahájením běhu programu). Díky tomu lze šrouby využít jako referenční strukturu při odhadu trajektorie elektrod. Inkluze šroubů ve výsledné masce je tedy podmínkou pro správný průběh algoritmu.
Abych toto dodržel, vytvořil jsem kolem každého vstupního bodu kulovou oblast zájmu s poloměrem jednoho centimetru. Problém nastává v momentě, kdy se v této oblasti zájmu nachází více elementů, než sledovaný šroub. Jedná se zejména o dráty přivedené k elektrodám, nebo části jiných šroubů, nacházejících se v bezprostředním okolí. Tyto struktury je třeba odstranit, jinak by později interferovali s žádaným průběhem algoritmu. Tento problém řeším tak, že prvotní verzí kulového okolí přenásobím snímek CT, čímž izoluji sledovanou oblast. Ze struktur přítomných v této oblasti vyberu tu největší (sledovaný šroub) a aktualizuji oblast zájmu tak, aby obsahovala pouze tuto strukturu. Toto provedu pro všechna okolí šroubů, upravená okolí sečtu, a tím získám druhou část masky. Tento krok stojí na knihovně *scikit-image* konkrétně na třídě *measure*, jejíž metoda *label* označí jednotlivé struktury a funkci *bincount* knihovny *numpy*, která počítá jejich velikost.
Nakonec již stačí obě masky sečíst a aplikovat je na CT snímek. Vzhledem k tomu že masky jsou binárního rázu, aplikace masky jednoduše znamená přenásobení CT snímku maskou. Tímto je ukončena pomyslná první část algoritmu.

3.4 Implementace algoritmu - druhá část

V druhé části algoritmu dochází k matematické aproximaci elektrod a následného využití této aproximace k detekci jednotlivých kontaktů.

3.4.1 Odhad trajektorie, prokládání polynomem



Obrázek 3.7. Operace odhadu trajektorie v kontextu druhé části algoritmu.

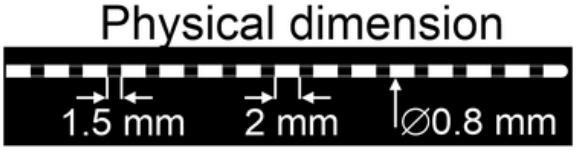
Výstupem první části algoritmu je upravený snímek CT, který obsahuje pouze části elektrod nacházející se v mozku pacienta a šrouby, kterými elektrody do pacientovy lebky vstupují. Další krok algoritmu detekuje jednotlivé elektrody. Samotná detekce ovšem není úplně snadná, neboť elektrody jsou z flexibilního materiálu a často dochází k jejich zvlnění, mírnému zkroucení nebo obecně odchýlení od přímé plánované trajektorie. Proto je k detekci elektrod využit postup, který nejprve identifikuje jednotlivé elektrody pomocí klasifikátoru založeném na Gaussově směsovém modelu (GMM), který dokáže správně označit skupiny voxelů odpovídající konkrétním elektrodám a umožní tak budoucí detekci jednotlivých kontaktů samostatně pro každou elektrodu. Pro zajištění úspěšného klasifikace GMM je potřeba při inicializaci předat algoritmu alespoň přibližné trajektorie elektrod stejně jako přibližný začátek a konec každé elektrody.
Začátkem elektrody je myšlen její první kontakt (KH), tedy hrot. Koncem elektrody (KP) se pak myslí poslední kontakt, tedy kontakt který je nejblíže lebeční kosti. Při znalosti posledního kontaktu a trajektorie $\vec{V}$ lze jednoduše určit pozici prvního kontaktu jako

$$KH = KP + L\vec{V} \tag{5}$$

L představuje délku elektrody kterou získáme jako

$$L = 2N + 1.5(N - 1) \tag{6}$$

kde N přestavuje počet kontaktů. Rovnice platí neboť každý kontakt je dva milimetry široký a mezera mezi kontakty je 1.5 milimetru.



Obrázek 3.8. Fyzické rozměry elektrod. Zdroj: [8]

Konec elektrody lze ztotožnit se vstupním bodem zadaným uživatelem (v blízkosti šroubu). Jediné, co je tedy potřeba získat, je trajektorie $\vec{V}$. Tuto trajektorii lze v duchu aproximace zaměnit za trajektorii šroubu. Výsledná trajektorie sice nebude odpovídat zcela stoprocentně skutečné situaci, ale velkou výhodou je, že výpočet trajektorie šroubu

je velmi robustní. Využívá faktu, že jsme v jednom z předchozích kroků byly všechny šrouby izolovány a následně uloženy do jedné matice a jsou tedy snadno k dispozici.
Při výpočtu trajektorie se postupuje následovně. Z pole, které obsahuje pouze izolovaný šroub se extrahují souřadnice x, y, z nenulových voxelů. V tomto kontextu odpovídají souřadnice x,y,z konvenci indexů i,j,k z IJK prostoru. Z těchto souřadnic se zvolí ta s nejvyšším rozptylem jako referenční. Následuje projekce na dvě roviny. Obě jsou definované referenční souřadnicí a pak jednou ze zbývajících souřadnic, tak aby byl pokrytý celý prostor. Tedy v případě, kdy je souřadnice s nejvyšším rozptylem souřadnice x, projekce budou provedeny na roviny xy a xz. Obě z těchto projekcí se proloží polynomem prvního řádu. V mé implementaci k tomu používám funkci *polyfit* z knihovny *numpy*, která proložení provádí na základě metody nejmenších čtverců, tedy minimalizuje chybu

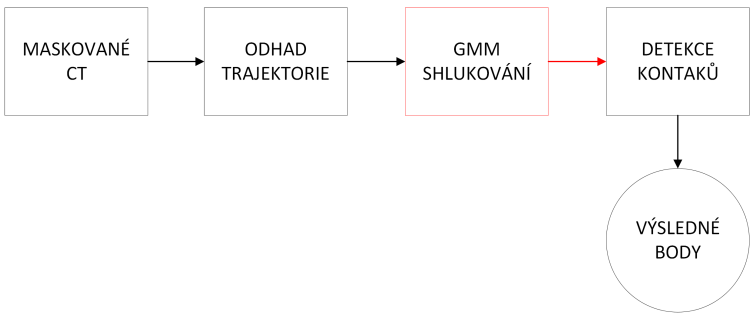
$$E = \sum_{j=0}^{k} |p(x_j) - y_j|^2 \tag{7}$$

kde $k+1$ je počet prvkům $p(x_j$ je hodnota polynomu evaluovaná pro x_j a y_j je skutečná hodnota v bodě x_j. Souřadnice x a y se zde vztahují ke konkrétní projekci. Funkce polyfit ovšem dovoluje nepovinný argument specifikující váhu, čehož využívám a váhuji každý voxel jeho intenzitou dle HU. To má za následek robustnější aproximaci, protože maximum intenzity se nachází v ose šroubu a je nižší na okrajích, kde vznikají artefakty rozptylu rentgenového záření na kovech. Váhovaná chybová funkce vypadá takto

$$E = \sum_{j=0}^{k} w_j |p(x_j) - y_j|^2 \tag{8}$$

kde w_j odpovídá intenzitě j-tého voxelu. [22]
Pokud budeme opět uvažovat jako refenční souřadnici x, výsledná křivka bude definována jako $x, y = P_{x-y}(x), z = P_{x-z}(x)$, kde $P_{x-y}(x), P_{x-z}(x)$ představují polynomy získané proložením dané roviny funkcí polyfit. Takto získaná křivka odpovídá středové ose šroubu.
Směrový vektor $\vec{V}$ se pak získá jednoduchým dosazením dvou různých hodnot x do předpisu křivky. Tento vektor stačí dosadit do rovnice (5) čímž se získá aproximace pozice hrotu elektrody. Nyní je vše připraveno pro úspěšnou inicializace algoritmu GMM.

3.4.2 Clustering pomocí Gaussova směsového modelu



Obrázek 3.9. Aplikace GMM algoritmu v kontextu druhé části algoritmu.

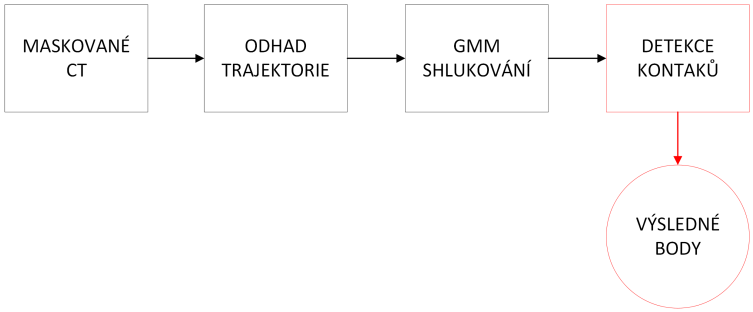
Gaussův směsový model je statistický model, který předpokládá data složená z několika různých gaussovských rozdělení s vlastními parametry. Jedná se o výkonný nástroj

pro analýzu a modelování dat, který je používaný hlavně pro svou flexibilitu a robustnost vůči různým tvarům shluků [23]. V této aplikaci je GMM využito pro detekci skupin voxelů, které společně tvoří jednotlivé elektrody.
K výpočtu GMM používám implementaci z knihovny *scikit-learn.mixture*. GMM není inicializováno náhodně, ale jako sada protáhlých normálních rozdělení přibližně kopírujících trajektorii elektrod. Jejichž počet odpovídá počtu hledaných elektrod. Každé z rozdělení je definováno těmito parametry:

- střední hodnota μ, která v rámci této aplikace odpovídá odhadnutému středu elektrody (MP). Ten se určí využitím bodů získaných v předchozím kroku jako $MP = (KH + KP)/2$
- kovarianční matice Σ, ta se získá aplikováním funkce *cov* z knihovny *numpy* na matici jejíž první řádek tvoří bod KH a druhý řádek bod KP
- váha π kterou je nastavena na hodnotu 1/N, kde N je počet elektrod. Všechny elektrody tedy mají stejnou váhu.

Jakmile jsou spočítány všechny parametry všech rozdělení, tj. je vytvořen list středních hodnot, list kovarianční matic a list vah, je možné GMM inicializovat. Kromě zmíněných parametrů je třeba při inicializaci ještě specifikovat počet rozdělení, ten samozřejmě odpovídá počtu elektrod, a typ kovariance, který je potřeba nastavit na `full` aby algoritmus uvažoval pro každý prvek vlastní kovarianční matici. Ještě je potřeba zmínit, že místo samotné kovarianční matice se jako vstup GMM používá její inverze. K jejímu výpočtu jsem opět použil knihovnu *numpy*, tentokrát funkci *inv* třídy *linalg*. Jakmile je na základě těchto parametrů vytvořen model, stačí jej aplikovat na relevantní data, tedy nenulové souřadnice vymaskovaného snímku CT. Po fitu modelu na data se zavolá funkce *predict*, která označkuje veškeré voxely na základě toho, ke kterému shluku neboli elektrodě patří, a tím je detekce jednotlivých elektrod ukončená.

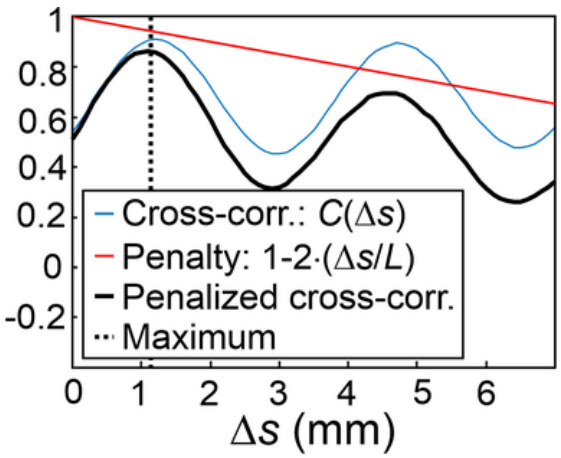
3.4.3 Detekce kontaktů



Obrázek 3.10. Operace detekce kontaktů v kontextu druhé části algoritmu.

Posledním krokem algoritmu je detekce jednotlivých kontaktů na elektrodách. Elektrody jsou separované z předchozího kroku algoritmu. Detekci kontaktů zde popíšu pro jednu elektrodu, nicméně tento proces se analogicky zopakuje pro každou z elektrod.
V prvním kroku se elektroda matematicky aproximuje za použití úplně stejného procesu, kterým byla vytvořena středová osa šroubu v kapitole 3.4.1. Jediný rozdíl je ve stupni polynomu. Elektroda může být mírně zkroucená (ale stále izomorfní), proto je pro její aproximaci použit vyšší řád polynomu. Já v mé implementaci používám polynom stupně tři. Ze získaného spojitého předpisu křivky vytvořím diskrétní reprezentaci křivky (1000 bodů). Při stanovování hranic se opět odkazuji na vstupní body. Jako první bod je vybrán nejvzdálenější bod od vstupního bodu, poslední bod je pak ten

nejbližší vstupnímu bodu. Tímto způsobem je vytvořeno tisíc bodů, které se všechny nacházejí na ose elektrody.
Po vytvoření diskrétní osy začne proces hledání bodů na této ose, které nejlépe aproximují pozice kontaktů. V první iteraci procesu se jako bod odpovídající prvnímu kontaktu zvolí úplně první bod na ose elektrody. Jako druhý bod se volí bod vzdálený 3.5 milimetrů od prvního vybraného bodu ve směru osy. Vzdálenost 3.5 milimetrů se volí protože, je to vzdálenost mezi středy dvou sousedních kontaktů. Zbytek bodů se pak volí analogicky, vždy se hledá bod vzdálený 3.5 milimetrů od naposledy vybraného bodu. Toto se opakuje dokud počet zvolených bodů neodpovídá počtu kontaktů na elektrodě.
Jakmile je první set bodů vytvořený, testuje se jak přesná tato aproximace kontaktů je. To probíhá tak, že se vytvoří prázdná kopie pole obsahujícího elektrodu CT a v této kopii se na pozicích specifikovaných testovaným setem bodů vytvoří kulové gaussiány se směrodatnou odchylkou jeden milimetr. Takto vytvořený obraz se následně koreluje s obrazem elektrody v CT. Výslednou hodnotu korelace si ukládám. Pro výpočet korelace používám funkci *corrcoef* knihovny *numpy*.
Toto opakuji ve forcyklu s tím, že při každé iteraci vybrané body o kousek posunu v ose elektrody směrem ke konci elektrody (tj. ven z lebky). Předtím, než se ověří která sada vybraných bodů v závislosti na posunu měla nejvyšší korelaci s CT, se na hodnoty korelace aplikuje penalizační funkce. Tato funkce s předpisem $1 - 2(\delta s/L)$, kde δs je posun v mm a L délka elektrody v mm přiřazuje sadám bodů vzdálenějších od hrotu elektrody nižší váhu. Děje se to proto, aby bylo zajištěno pokládání prvního bodu na první kontakt i v případech, kdy je první kontakt zkreslen artefakty [8].
Sada bodů, která má po přenásobení hodnot korelací penalizační funkcí nejvyšší korelaci je vybrána jako finální, tedy tyto body přímo odpovídají souřadnicím kontaktů. Tyto body pak stačí převést z IJK do RAS souřadnicového systému a vytvořit z nich `vtkMRMLMarkupsFiducialNode`.



Obrázek 3.11. Graf korelace mezi modelem a snímkem CT. Modrá křivka - korelace, černá křivka - korelace po penalizaci, červená křivka - penalizační funkce. Převzato z [8]

3.4.4 Grafické rozhraní

Při implementaci grafického rozhraní byl dbán důraz hlavně na funkčnost. Po estetické stránce se aplikace drží stylu použitého v ostatních modulech Sliceru.
Celé grafické rozhraní stojí na knihovnách *PyQt* a *ctk* a využívá hlavně rozbalovacích tlačítek, *comboboxů* a obyčejných tlačítek.

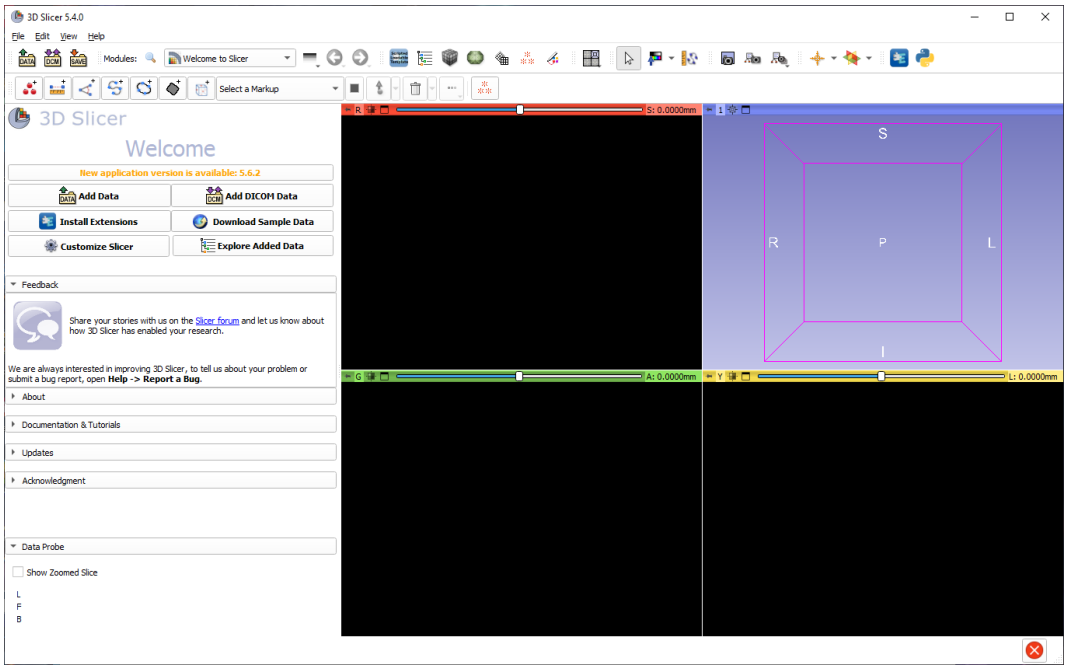
Veškerý text v rámci rozšíření je psán anglicky, stejně jako je tomu ve zbytku aplikace Slicer.
Veškeré prvky jsou bílé s černými rámečky a černým textem. Pokud má být některý z prvků zvýrazněný je použito tučné písmo.
Grafické rozhraní je jak z pohledu grafického, tak z pohledu implementačního poměrně strohé a standardní, proto není třeba detailněji se jí detailněji zabývat.

Kapitola 4
Výsledná aplikace

V této kapitole popíši funkci jednotlivých prvků výsledné aplikace a to jak s nimi pracovat.

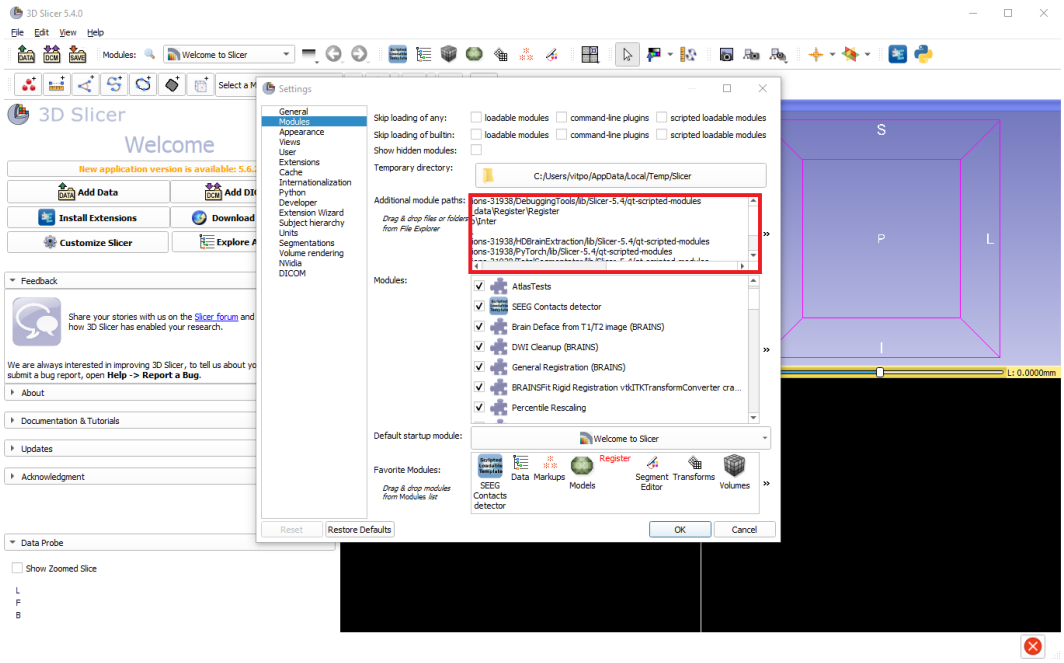
4.1 Instalace Aplikace

Pro použití aplikace musí mít uživatel nainstalovanou poslední verzi (5.6.2) 3D Sliceru (volně k dispozici na `https://download.slicer.org/`). Pokud má uživatel Slicer nainstalovaný a má stažený zdrojový soubor aplikace, může pokračovat s instalací.



Obrázek 4.1. Domovská stránka aplikace 3D Slicer.

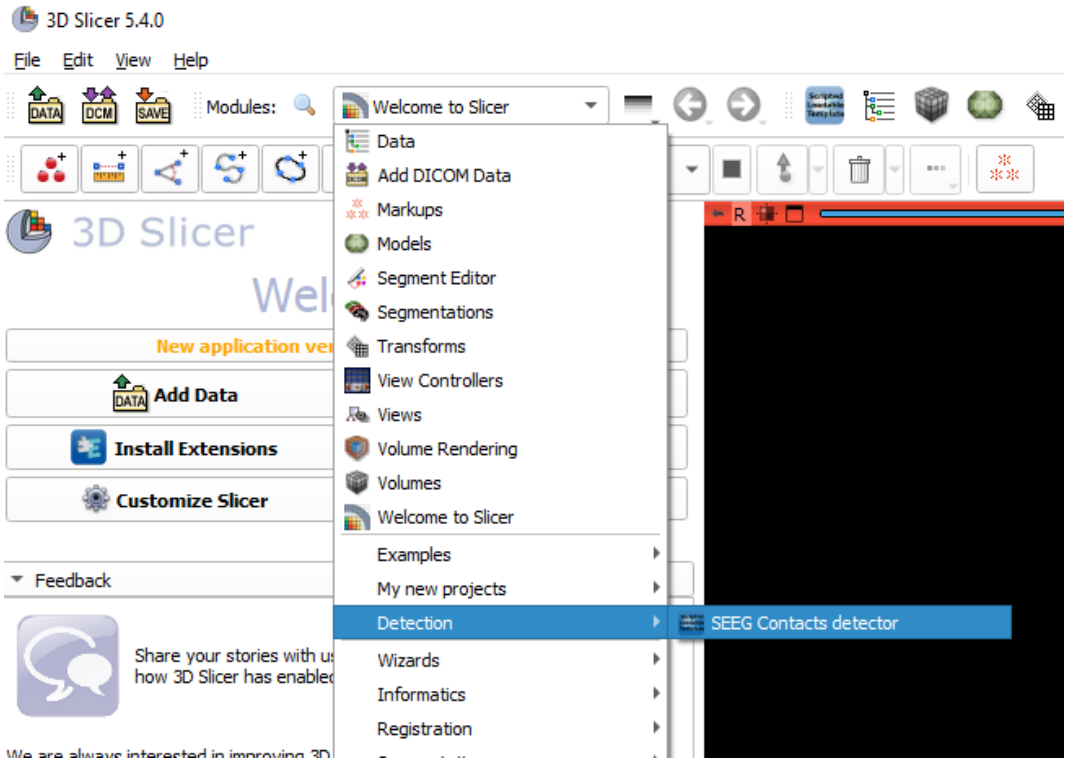
Prvním krokem instalace rozšíření je spustit aplikaci Slicer, následně kliknou v horní liště programu na záložku `Edit` a vybrat možnost `Application settings`. Tím se otevře okno nastavení. V tomto okně je třeba překliknout na záložku `Modules` nacházející se v liště na pravé straně. V tomto okně je k dispozici oblast popsaná textem `Drag & drop files or folders from File Explorer`. Do této oblasti stačí přetáhnout extrahovanou složku s rozšířením a restartovat Slicer. Při dalším spuštění bude rozšíření k dispozici.



Obrázek 4.2. Okno Settings se zvolenou záložkou Modules. Oblast, do které se má přetáhnou složka obsahující soubory rozšíření je zvýrazněna červeně.

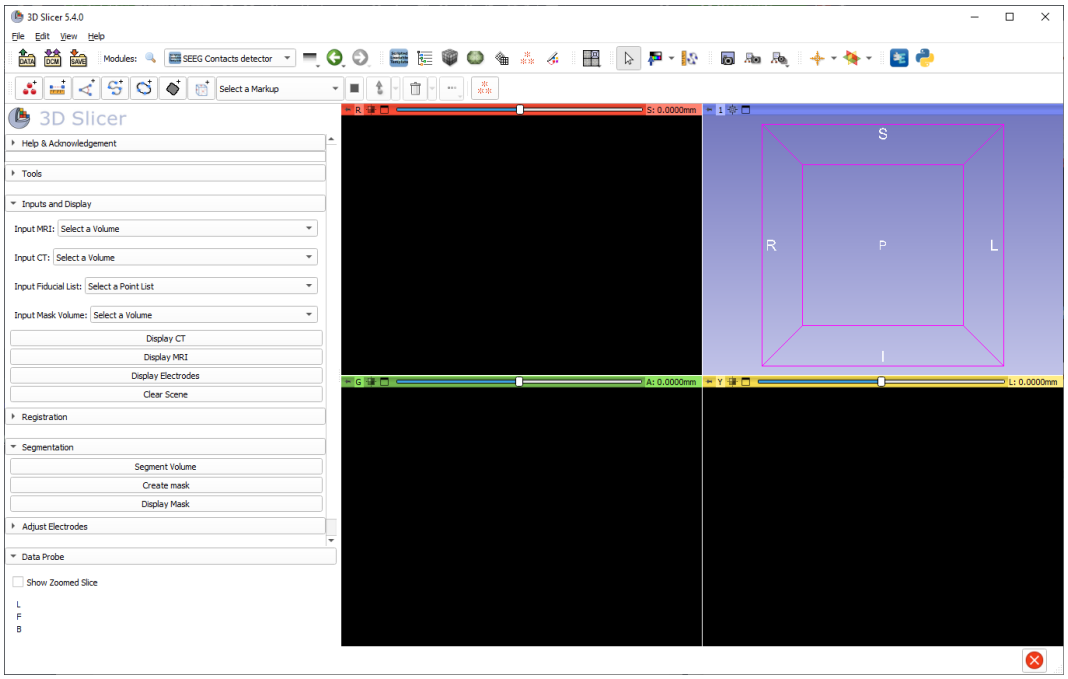
4.2 Spuštění rozšíření

Po instalaci je rozšíření k dispozici v selektoru modulů pod záložkou `Detection` jako `SEEG Contacts Detector`



Obrázek 4.3. Přístup k rozšíření skrze selektor modulů.

Po zvolení této možnosti se uživatel dostane ke grafickému rozhraní rozšíření.



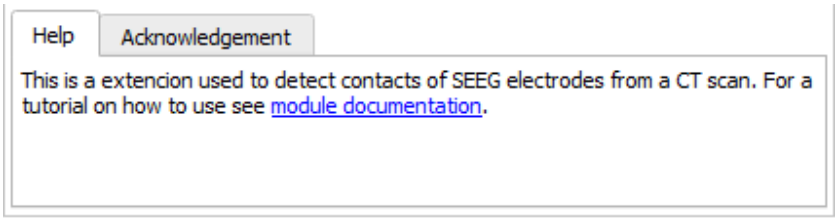
Obrázek 4.4. Uživatelské rozhraní rozšíření.

4.3 Prvky uživatelského rozhraní

V této kapitole popíšu všechny prvky uživatelského rozhraní a jejich použití. Základní strukturou uživatelského prostředí jsou rozbalovací tlačítka, někdy v textu také označeny jako karty. Zde je po jednom představím.

4.3.1 Help & Acknowledgement

Tlačítko `Help & Acknowledgement` je standartní součástí každého modulu aplikace Slicer. Osahuje záložku Help, ve které je k dispozici odkaz na návod pro použití modulu a záložku Acknowledgement, kde se nachází odkazy na použité technologie a poděkování jejich autorům.



Obrázek 4.5. Rozbalovací tlačítko `Help & Acknowledgement`.

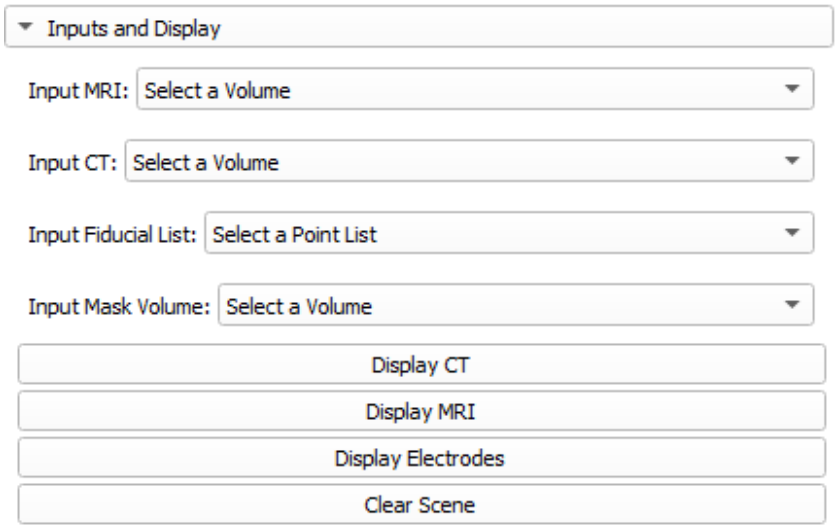
4.3.2 Tools

V kartě `Tools` se nachází tlačítko `Restart`, které restartuje aplikaci Slicer a tlačítko `Exit`, které aplikaci jednoduše ukončí. Při použití těchto tlačítek musí uživatel dbát na to, že po jejich stlačení nevyskočí žádné další okno, či upozornění, zvolená akce se jednoduše provede. To může vést ke ztrátě neuložených dat.

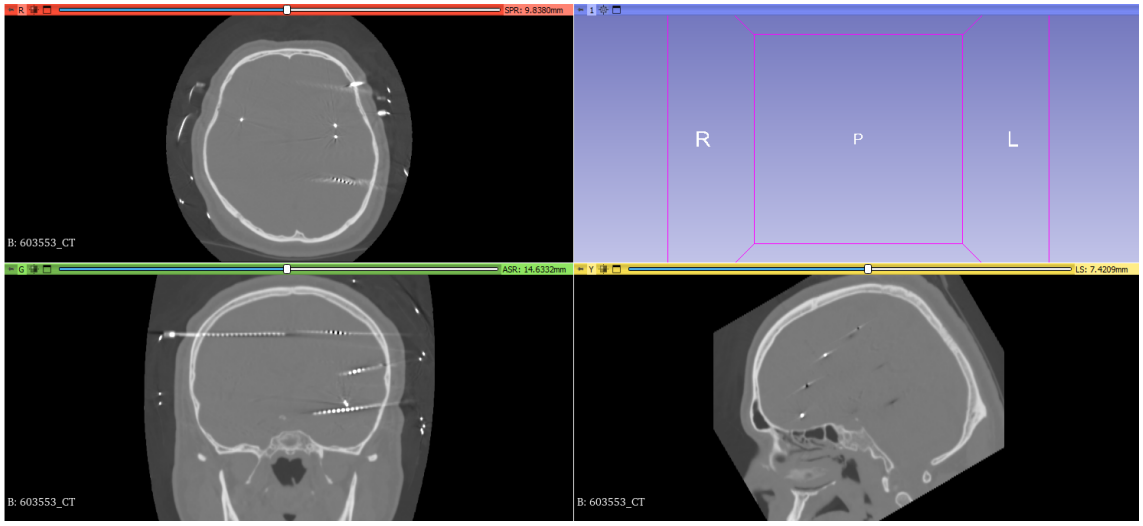
Obrázek 4.6. Rozbalovací tlačítko `Tools`.

4.3.3 Inputs and Display

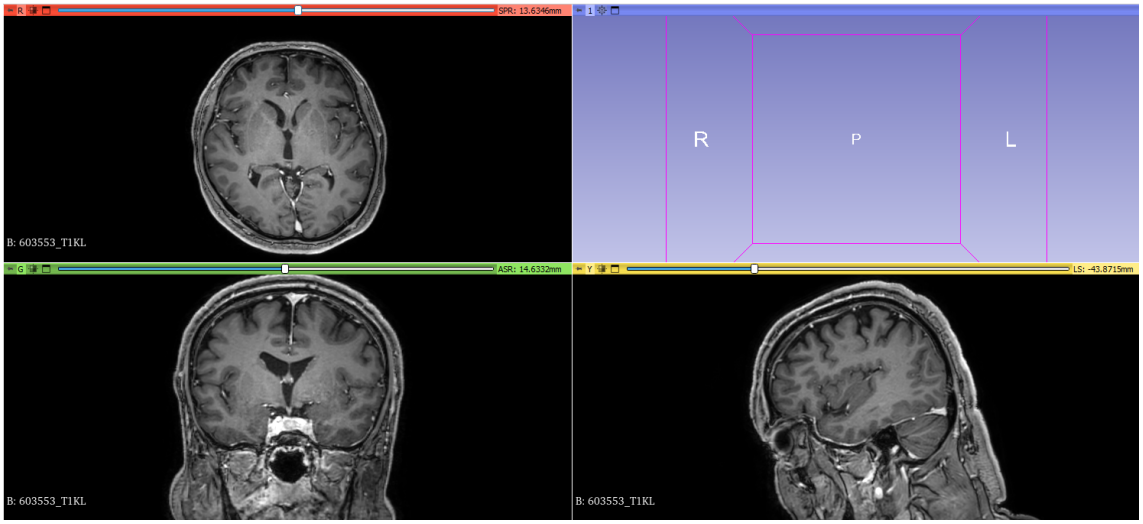
Karta `Inputs and Display` obsahuje funkcionalitu kritickou k běhu programu.
První čtyři prvky uvnitř této karty jsou selektory. V selektoru `Input MRI` uživatel specifikuje který z vložených snímků má fungovat jako vstupní snímek magnetické rezonance (MRI). U MRI se předpokládá T1 váhování. Analogicky pak v druhém selektoru `Input CT` volí vstupní snímek výpočetní tomografie (CT).
V selektoru `Input Fiducial List` uživatel vybere list obsahující body, které specifikují pozice šroubů a délku elektrod. Jestliže takový list uživatel nemá, musí si jej vytvořit. Jak toho dosáhnout je popsáno v další podkapitole.
Poslední selektor nabízí uživateli vložit vstupní masku. Maska by měla specifikovat oblast mozku, nic více. Tuto možnost uživatel volí, pokud má již předem segmentovaný snímek MRI a přeje si tuto část algoritmu v rámci tohoto modulu přeskočit. Tento vstup je tedy na rozdíl od tří předešlých volitelný.
Tlačítka `Display CT` a `Display MRI` slouží k zobrazení vstupních dat v grafických oknech Sliceru.
Tlačítko `Display Electrodes` zobrazí prahovaný snímek CT, tak aby šli vidět pouze elektrody a využije k tomu i okno určené pro zobrazování trojrozměrných struktur.
Poslední Tlačítko `Clear Scene` vyčistí scénu a vrátí ji tak do původního stavu.



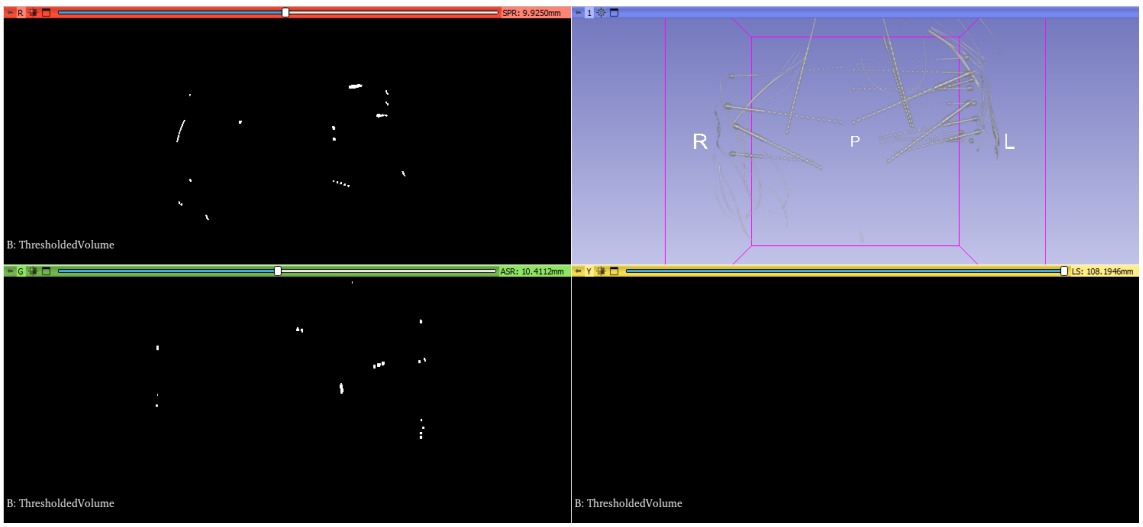
Obrázek 4.7. Rozbalovací tlačítko `Inputs and Display`



Obrázek 4.8. Grafický výstup tlačítka `Display CT`.



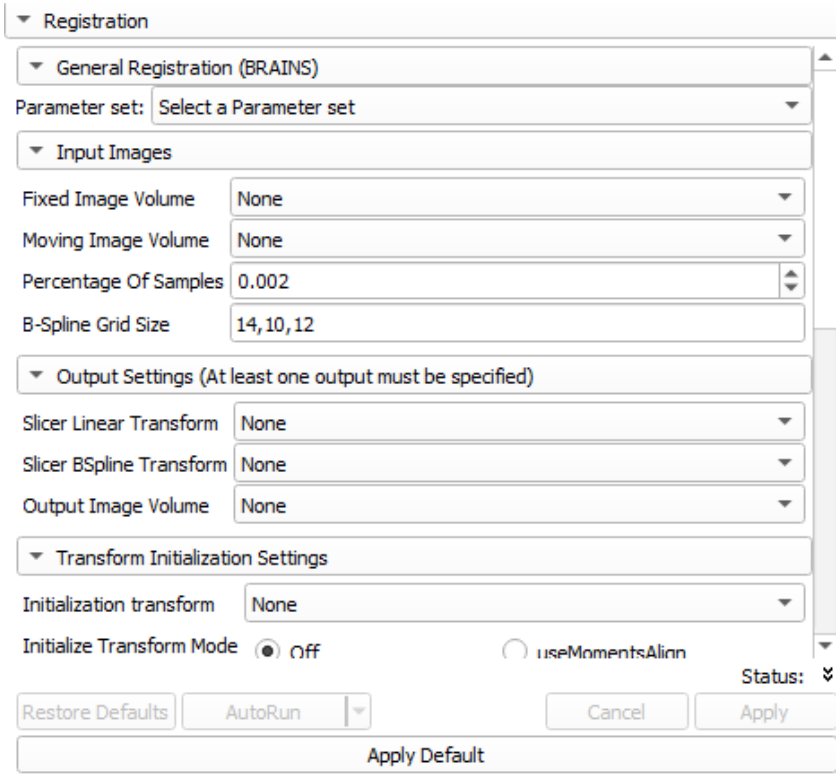
Obrázek 4.9. Grafický výstup tlačítka `Display MRI`.



Obrázek 4.10. Grafický výstup tlačítka `Display Electrodes`.

4.3.4 Registration

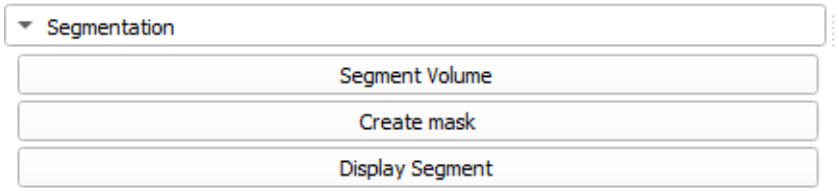
Rozbalovací tlačítko `Registration` obsahuje dva hlavní prvky. Prvním je reprezentace defaultního modulu používaného na registraci ve Sliceru, tedy modulu BRAINSFit. Skrze tento prvek má uživatel plný přístup k funkcionalitě tohoto modulu.
Pokud uživatel nemá zájem o vlastní specifikaci vstupních parametrů registrace, má k dispozici tlačítko `Apply default`, které aplikuje registraci s defaultními parametry blíže specifikovanými v kapitole zabývající se implementací registrace.



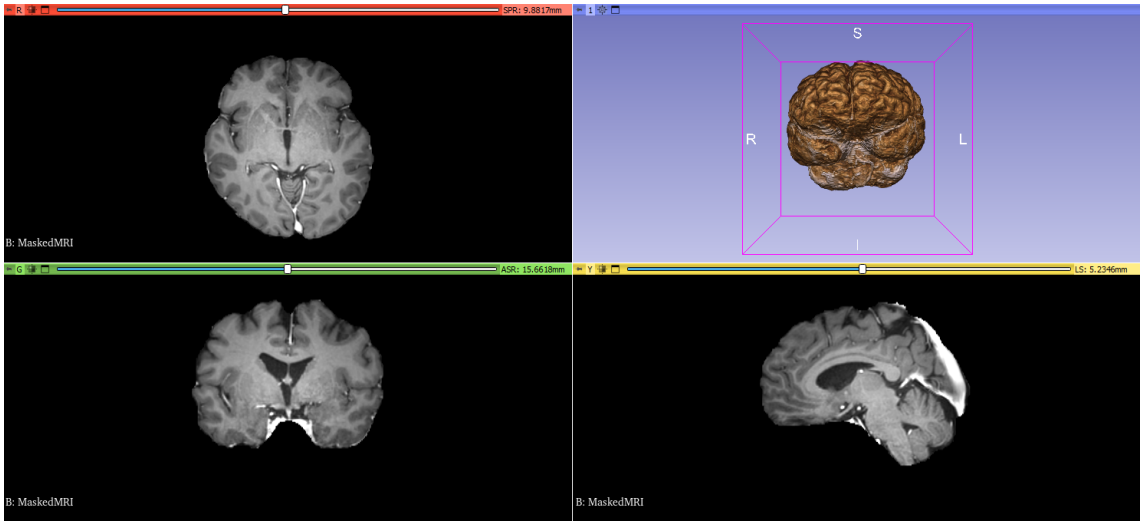
Obrázek 4.11. Rozbalovací tlačítko `Registration`.

4.3.5 Segmentation

Rozbalovací tlačítko `Segmentation` obsahuje tři tlačítka nabízející následující funkce.
Tlačítko `Segment Volume` zajistí segmentaci oblasti mozku ze vstupního MRI.
Tlačítko `Display Segment` zobrazí vysegmentovanou oblast.
Tlačítko `Create Mask` vytvoří masku oblasti zájmu, specifikovanou v kapitole implementace



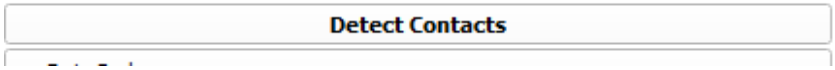
Obrázek 4.12. Rozbalovací tlačítko `Segmentation`.



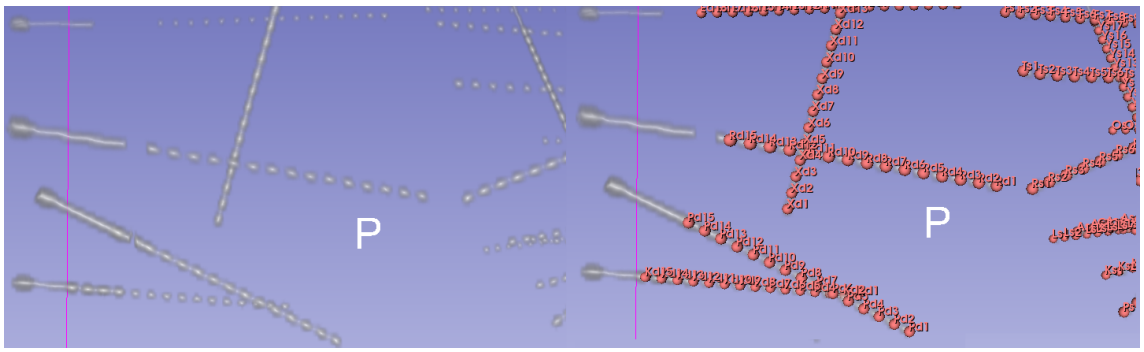
Obrázek 4.13. . Grafický výstup tlačítka `Display Segment`

4.3.6 Detect Contacts

Tlačítko `Detect Contacts` je jediné tlačítko nenacházející se uvnitř rozbalovacího menu. Je to proto aby vždy, bylo viditelné. Toto tlačítko totiž spouští samotný detekční algoritmus a svým způsobem je tak nejdůležitějším prvkem rozšíření.
Po jeho stisknutí se automaticky zobrazí vygenerované body.



Obrázek 4.14. Tlačítko `Detect Contacts`

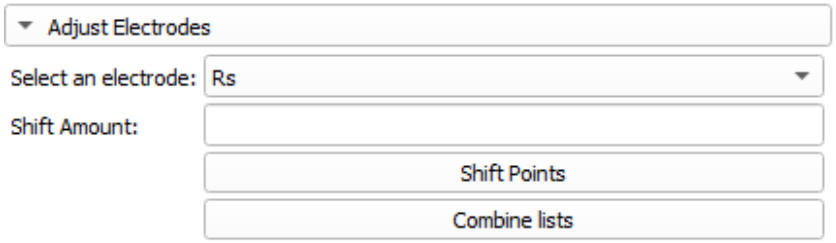


Obrázek 4.15. Scéna před a po spuštění detekčního algoritmu

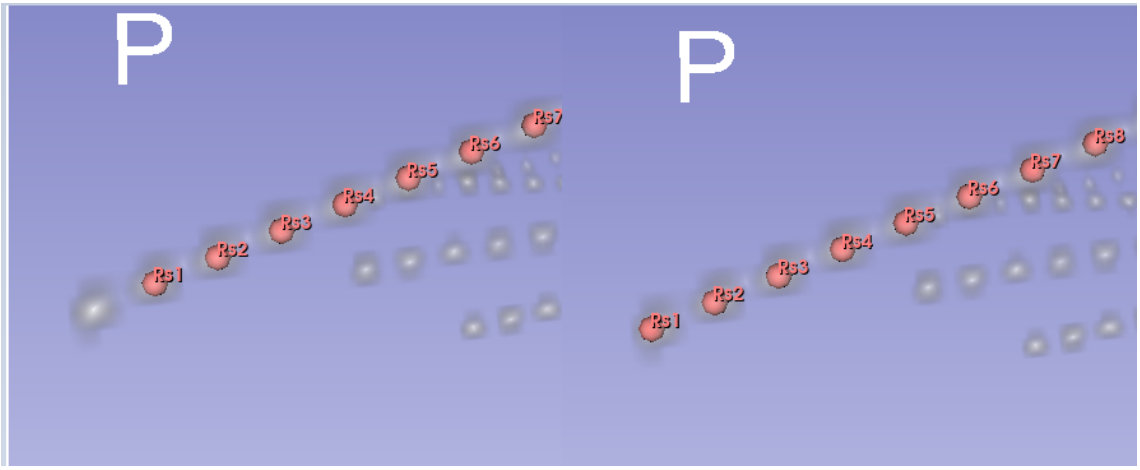
4.3.7 Adjust Electrodes

Posledním prvkem grafického rozhraní je karta `Adjust Electrodes`. V této kartě má uživatel možnost zvolit jednu z detekovaných elektrod a aplikovat na ni posun v případě, že algoritmus chybně detekoval např. první kontakt elektrody. Tím je umožněna případná manuální korekce nepřesné detekce.
Uživatel nejprve zvolí na kterou elektrodu chce posun aplikovat. Elektrody jsou pojmenovány podle příslušných vstupních bodů.
Hodnotu posunu uživatel nastaví manuálním zadáním čísla do okna označeného `Shift amount`. Kladná čísla posunou body směrem od středu mozku, záporná směrem ke středu mozku. Hodnota posunu není v milimetrech, uživatel musí experimentálně

přijít na to, jaká hodnota je pro něj nejvhodnější. Jako první doporučuji zkusit hodnotu deset a řídit se, podle výsledku. Tlačítko `Shift points` pak posun na body aplikuje. Výsledek je instantě viditelný v grafickém okně.Nicméně, v budoucí verzi programu bude vhodné nahradit hodnotu v mm.
Poslední tlačítko přítomné v této kartě je `Combine lists`. Po stisknutí tohoto tlačítka se všechny listy reprezentující jednotlivé elektrody spojí do jednoho listu, který pak může uživatel snadno uložit využitím `SAVE` funkce 3D Sliceru. Listy použité pro vytvoření finálního listu jsou následně ze scény odstraněny, aby scéna nebyla přeplněná. To znamená, že po zkombinování listů již není možné aplikovat posun na jednotlivé elektrody.



Obrázek 4.16. Rozbalovací tlačítko `Adjust Electrodes`.



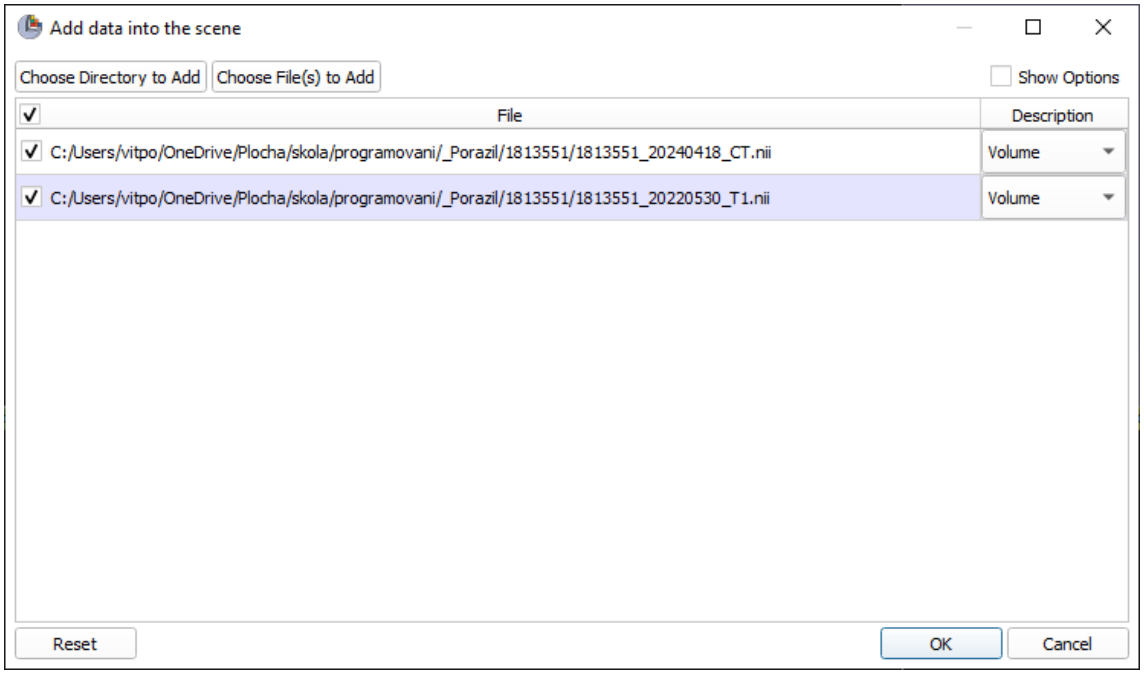
Obrázek 4.17. Body na elektrodě před a po aplikování posunu

4.4 Použití aplikace

V této podkapitole je vysvětleno jak má uživatel pracovat s aplikací, aby získal požadovaný výsledek. Předpokládá se, že uživatel je již seznámený s jednotlivými prkvy rozšíření a není proto nutné vše vysvětlovat do nejmenších detailů.

4.4.1 Nahrání dat

Postup nahrání dat potřebných pro běh program je velmi snadný. Pokud jsou data v Slicerem podporovaném formátu (.nii, .nii.gz, .nrrd a další), stačí soubory obsahující CT a MRI snímky jednoduše přetáhnou do prostředí Sliceru a výběr potvrdit.

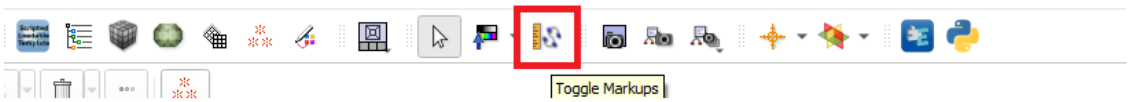


Obrázek 4.18. Obrazovka potvrzující nahrání dat do Sliceru

Pak již stačí aby uživatel v sekci `Inputs and Display` správně navolil, který snímek má sloužit jako vstupní magnetická rezonance a který jako vstupní snímek CT.

4.4.2 Vytvoření vstupního listu bodů

V dalším kroce musí uživatel specifikovat list bodů, které mají být použity jako vstupní body pro algoritmus. Uživatel má tedy za úkol označit bodem každý ze šroubů, kterým vstupuje elektroda do oblasti mozku a každý z těchto bodů pojmenovat ve formátu `JménoElektrody-Číslo`, kde jméno elektrody jsou typicky dvě písmena a číslo vyjadřuje počet kontaktů, které tato elektroda má. Bod pojmenovaný `Rs-15` tak definuje elektrodu Rs, která má patnáct kontaktů.
Uživatel má pro tento úkol k dispozici tlačítko `Display Electrodes`, které veškeré šrouby a elektrody zobrazí, aby k nim měl uživatel snadný přístup.
Samotné body by pak uživatel měl vytvořit za pomoci nástroje na pokládaní bodů, který nalezne v horní liště programu (viz obrázek ??). Pokládané body budou automaticky pojmenovány, uživatel je proto musí po položení ručně přejmenovat. Po najetí myší na bod a stisknutí pravého tlačítka se zobrazí menu, ve kterém uživatel vybere položku `Rename` a bod přejmenuje.
Jakmile uživatel označí všechny šrouby, musí nově vytvořený list zvolit jako vstupní v sekci `Inputs and Display`. Pokud uživatel list manuálně nepřejmenoval, najde list pod jménem `F`.



Obrázek 4.19. Červeně zvýrazněné tlačítko otevírající lištu umožňující pokládání bodů

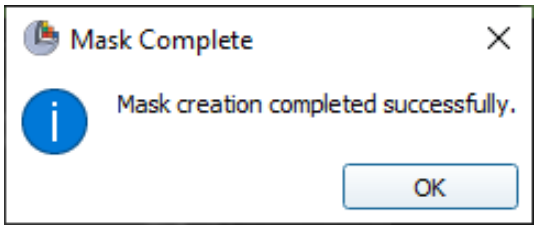
Obrázek 4.20. Lišta umožňující pokládání bodů.

4.4.3 Registrace snímků

Jestliže vstupní snímky nejsou registrovány, musí je uživatel registrovat. Stačí aby v kartě `Register` klikl na tlačítko `Apply Default`, případně aby využil vloženého modulu BRAINSFit.

4.4.4 Vytvoření masky

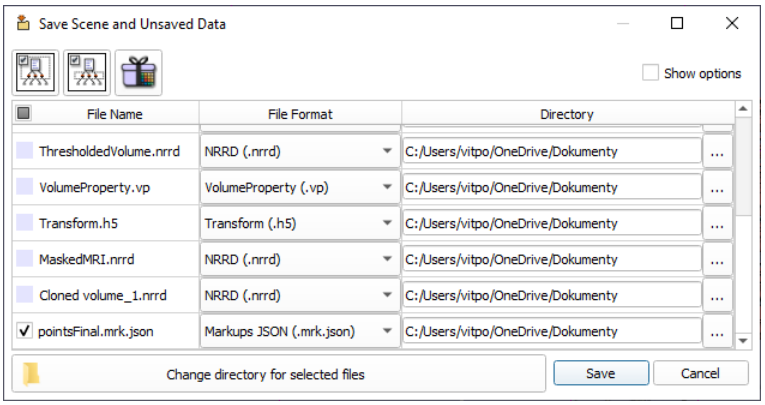
V dalším kroku musí uživatel zajistit vytvoření masky. Toho dosáhne jednoduše tak, že v sekci `Segmentation` zvolí možnost `Segment Volume` a počká, než se vysegmentuje oblast mozku z MRI snímku. Aplikace oznámí ukončení výpočtu vyskakovacím oknem. Následně uživatel stiskne tlačítko `Create Mask` a jakmile vyskočí vyskakovací okno signalizující konec výpočtu, je maska hotová.



Obrázek 4.21. Vyskakovací okno oznamující úspěšnou segmentaci mozku

4.4.5 Detekce kontaktů

Pokud uživatel úspěšně splnil všechny předešlé kroky může přejít k samotné detekci kontaktů. Stačí aby stiskl tlačítko `Detect Contacts` a počkal, než algoritmus dokončí výpočet.
Pokud je po běhu algoritmu uživatel nepokojený s pozicemi bodů některé z elektrod, může je upravit v kartě `Adjust electrodes`, tak jak je vysvětleno v podkapitole výše. Alternativně taky může uživatel body posunout ručně potahnutím myší. Jakmile je uživatel s pozicemi bodů spokojen, může si je uložit buď po elektrodách, nebo jako jeden list. Ke sloučení listů slouží tlačítko `Combine lists` uvnitř karty `Adjust electodes` k uložení listu pak slouží tlačítko `SAVE` nacházející se v levém horním rohu aplikace Slicer. Jakmile uživatel výsledný list uloží, úspěšně splnil úlohu detekce kontaktů a může aplikace ukončit.

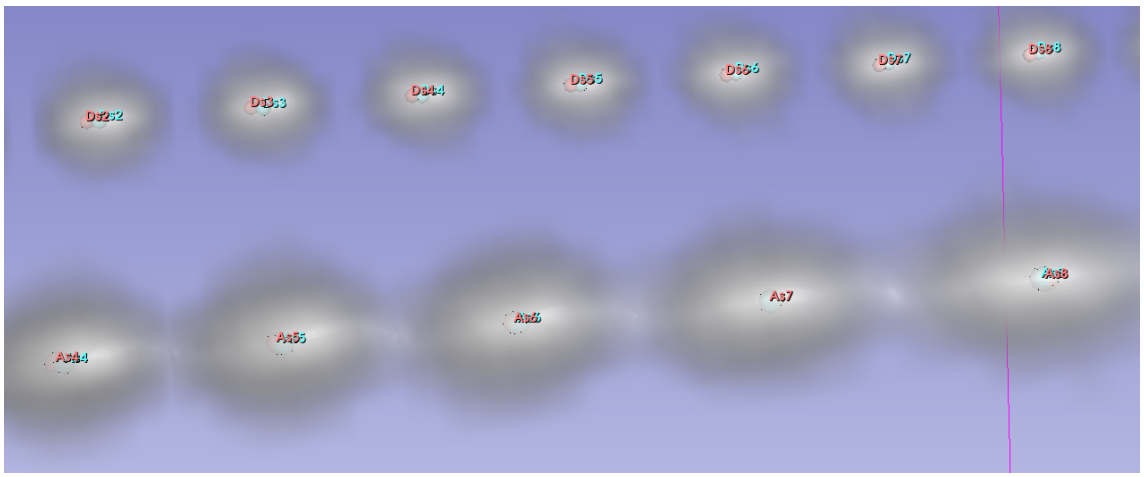


Obrázek 4.22. Okno pro pro ukládaní dat, které se zobrazí po stisknutí tlačítka `SAVE`

Kapitola 5
Testování aplikace

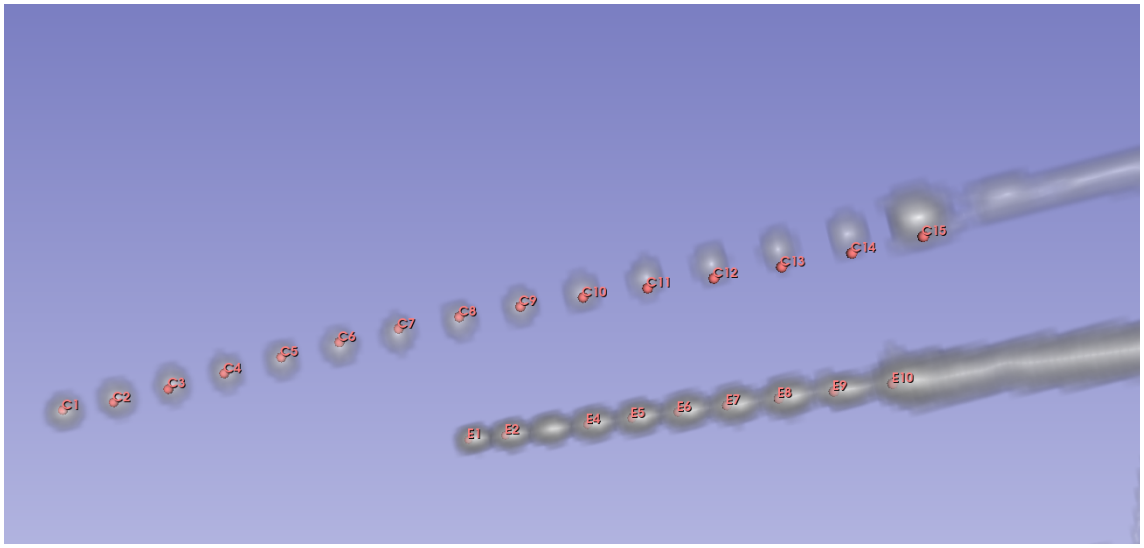
5.1 Testování přesnosti detekce

V této kapitole je otestována přesnost detekce kontaktů aplikací. Celkově proběhlo testování na šesti pacientech, u kterých byla detekce kontaktů již provedena použitím referenčního automatického algoritmu [8]. Pacienti měli dohromady implantovaných 81 elektrod s celkem 1071 kontakty.
Vytvořené body byli porovnány s referenčními ve čtyřech metrikách. U každého bodu byla změřena odchylka v ose x, y a z a celková euklidovská vzdálenost od referenčního bodu. Testování ukázalo, že průměrná odchylka v ose x byla $0.14 \pm 0.1mm$, v ose y $0.07 \pm 0.08mm$ a v ose z $0.09 \pm 0.09mm$. Body byly v průměru $0.21 \pm 0.10mm$ daleko od referenčních.
Vizuální analýza ukazuje, že můj algoritmus neidentifikuje středy kontaktů s takovou přesností, jako algoritmus referenční. Příčinou tohoto problému nejpravděpodobněji je, že při vytváření sad aproximačních bodů, tak jak je popsáno v kapitole 3.4.3 je odstup mezi jednotlivými testovanými sadami příliš velký a optimální poloha bodů je tak přeskočena. Pokud by se tento odstup zmenšil, přesnost algoritmu by se za cenu vyšší výpočetní náročnosti zlepšila.
Ani jednou nenastal případ, kdy by algoritmus některý z kontaktů označil špatně. Typický výsledek algoritmu je zobrazený na obrázku níže.



Obrázek 5.1. Elektrody As a Ds u čtvrtého pacienta, modré body referenční.

Nejhoršího výsledku dosáhl algoritmus na elektrodě `C` u pacienta číslo 5. Při vizuální analýze je patrné, že algoritmus správně označil kontakty patřící elektrodě, ale při aproximaci osy elektrody došlo k chybě. Z tohoto důvodu jsou body vychýleny.



Obrázek 5.2. Nejhorší výsledek algoritmu, elektroda C u pátého pacienta.

Žádné další chyby při vizuální kontrole odhaleny nebyly a to i v případě prvního pacienta, který byl považovaný na rizikového, kvůli přítomnosti dalších kovových předmětů ve oblasti hlavy.
Veškerá naměřená data jsou zobrazena v tabulkách níže. Pro přehlednost je vždy uvedena průměrná hodnota dané veličiny, místo aby byly vypsány hodnoty pro každý kontak zlvášť. V kontextu první tabulky tedy hodnota $d\bar{x}$ elektrody CA představuje aritmetický průměr oddchylek jednotlivých kontaktů v ose x od referenčních kontaktů.

Elektroda	B	C	D	E	F	G	H
$\mathrm{d}\bar{x}$	0.10	0.02	0.02	0.02	0.02	0.13	0.14
$\mathrm{d}\bar{y}$	0.01	0.02	0.05	0.06	0.01	0.05	0.05
$\mathrm{d}\bar{z}$	0.19	0.23	0.26	0.15	0.10	0.02	0.02
$\overline{vzdálenost}$	0.21	0.24	0.26	0.17	0.11	0.14	0.16

Elektroda	I	J	O	P	Q	R	S	T
$\mathrm{d}\bar{x}$	0.16	0.36	0.02	0.17	0.42	0.07	0.02	0.28
$\mathrm{d}\bar{y}$	0.07	0.10	0.06	0.03	0.07	0.06	0.06	0.24
$\mathrm{d}\bar{z}$	0.04	0.02	0.15	0.21	0.17	0.01	0.37	0.03
$\overline{vzdálenost}$	0.20	0.38	0.17	0.27	0.46	0.10	0.37	0.42

Tabulka 5.1. Výsledky testování na datech prvního pacienta

5.2 Testování funkčnosti aplikace

Testování funkčnosti aplikace proběhlo pasivně při získávání dat z aplikace pro testování přesnosti detekce.

5.3 Odhalené problémy

Během práce s aplikací bylo odhaleno několik chyb programu, které by se měli v další verzi opravit. Zde je jejich výčet.

Elektroda	As	Bs	Cd	Cs	Ds	Gs	Ks
$d\bar{x}$	0.14	0.19	0.34	0.08	0.21	0.10	0.08
$d\bar{y}$	0.01	0.07	0.09	0.02	0.03	0.01	0.49
$d\bar{z}$	0.09	0.01	0.02	0.10	0.03	0.01	0.03
$\overline{vzdálenost}$	0.17	0.21	0.36	0.14	0.21	0.10	0.50

Elektroda	Ld	Ls	Os	Ps	Rs	Ts	Ws
$d\bar{x}$	0.10	0.11	0.01	0.05	0.02	0.27	0.22
$d\bar{y}$	0.05	0.02	0.33	0.02	0.05	0.01	0.02
$d\bar{z}$	0.02	0.02	0.08	0.04	0.11	0.06	0.09
$\overline{vzdálenost}$	0.12	0.12	0.34	0.07	0.13	0.28	0.24

Tabulka 5.2. Výsledky testování na datech druhého pacienta

Elektroda	A	B	C	D	E	F
$d\bar{x}$	0.11	0.30	0.25	0.14	0.04	0.05
$d\bar{y}$	0.01	0.06	0.09	0.20	0.21	0.09
$d\bar{z}$	0.06	0.19	0.10	0.12	0.20	0.09
$\overline{vzdálenost}$	0.14	0.37	0.29	0.28	0.30	0.15

Elektroda	G	H	I	J	K	L
$d\bar{x}$	0.13	0.08	0.19	0.15	0.18	0.12
$d\bar{y}$	0.09	0.07	0.11	0.07	0.05	0.11
$d\bar{z}$	0.03	0.02	0.04	0.05	0.03	0.03
$\overline{vzdálenost}$	0.18	0.11	0.23	0.18	0.19	0.18

Tabulka 5.3. Výsledky testování na datech třetího pacienta

Elektroda	A	B	C	D	E	F
$d\bar{x}$	0.13	0.05	0.14	0.35	0.11	0.21
$d\bar{y}$	0.02	0.01	0.02	0.04	0.04	0.02
$d\bar{z}$	0.09	0.01	0.02	0.13	0.04	0.08
$\overline{vzdálenost}$	0.15	0.05	0.14	0.38	0.13	0.23

Elektroda	G	H	O	P	PC	S	T
$d\bar{x}$	0.09	0.07	0.22	0.10	0.10	0.21	0.15
$d\bar{y}$	0.05	0.02	0.05	0.01	0.01	0.05	0.03
$d\bar{z}$	0.05	0.03	0.02	0.04	0.02	0.06	0.01
$\overline{vzdálenost}$	0.12	0.08	0.23	0.11	0.11	0.22	0.16

Tabulka 5.4. Výsledky testování na datech čtvrtého pacienta

- Při instalaci rozšíření dojde k chybě při instalování knihoven, uživatel je musí doinstalovat ručně.

Elektroda	A	B	C	D	E	F
$\mathrm{d}\bar{x}$	0.28	0.12	0.11	0.07	0.29	0.29
$\mathrm{d}\bar{y}$	0.06	0.02	0.38	0.08	0.03	0.19
$\mathrm{d}\bar{z}$	0.02	0.02	0.28	0.02	0.13	0.06
$\overline{vzdálenost}$	0.29	0.12	0.50	0.11	0.32	0.40

Elektroda	P	Q	R	T	X	Y	Z
$\mathrm{d}\bar{x}$	0.17	0.09	0.26	0.33	0.02	0.03	0.02
$\mathrm{d}\bar{y}$	0.02	0.02	0.06	0.03	0.03	0.01	0.13
$\mathrm{d}\bar{z}$	0.11	0.04	0.12	0.16	0.31	0.13	0.22
$\overline{vzdálenost}$	0.21	0.10	0.30	0.36	0.31	0.13	0.26

Tabulka 5.5. Výsledky testování na datech pátého pacienta

Elektroda	CA	CG	FA	FM	J	Jd
$\mathrm{d}\bar{x}$	0.07	0.18	0.13	0.11	0.14	0.12
$\mathrm{d}\bar{y}$	0.02	0.05	0.01	0.03	0.07	0.02
$\mathrm{d}\bar{z}$	0.40	0.04	0.05	0.02	0.08	0.08
$\overline{vzdálenost}$	0.40	0.20	0.15	0.12	0.18	0.15
0.39						

Elektroda	K	L	M	N	OF	R	X
$\mathrm{d}\bar{x}$	0.08	0.21	0.09	0.16	0.09	0.01	0.01
$\mathrm{d}\bar{y}$	0.07	0.07	0.02	0.04	0.02	0.16	0.07
$\mathrm{d}\bar{z}$	0.14	0.04	0.03	0.03	0.04	0.45	0.14
$\overline{vzdálenost}$	0.18	0.23	0.10	0.17	0.10	0.48	0.16

Tabulka 5.6. Výsledky testování na datech šestého pacienta

- Pokud se před spuštěním rozšíření nejprve otevře modul General Registration, reprezentace tohoto modulu v rámci našeho rozšíření přestane fungovat a vůbec se nezobrazí. Toto je pravděpodobně způsobeno špatným nastavením spojení s 3D Slicerem.
- Při posouvání bodů v záporném směru body občas zmizí ze scény, pokud se body posunou druhým směrem, objeví se zpět.
- Vyskakovací okno signalizující konec operace vytvoření masky je identické s oknem signalizujícím konec operace Segmentace. To může k zmatení uživatele.

5.4 Náměty na změny

Zde je seznam věcí, které stojí za uváženou při vývoji další verze rozšíření.

- Výsledné body nejsou konzistentně pokládány do středu kontaktů. V kapitole testování je popsáno, že přesnější výsledek by trval déle spočítat, nicméně co nejpřesnější aproximace by nejspíš měla být priorita.
- Reprezentace modulu BRANSFit uvnitř rozšíření je zbytečná, protože pro jeho použití musí mít uživatel předchozí zkušenosti s prací s tímto modulem a neměl by tedy

problém použít jej i mimo rozšíření. Naopak pro uživatele, který s prací s modulem zkušenosti nemá tvoří zbytečný vizuální šum.

- Tlačítka `Segment Volume` a `Display Segment` jsou pojmenována velmi vágně, není na první pohled jasné, co je jejich úkol.
- Vytváření masky je výpočetně velmi náročné, bylo by vhodné prozkoumat zdali neexistuje způsob jak tento proces optimalizovat. To samé platí pro detekci jednotlivých kontaktů.
- V průběhu téměř každého výpočtu se aplikace jeví jako `zamrzlá`, tedy není vůbec responzivní. To působí dojmem, že během průběhu programu došlo k chybě. Tento problém by pravděpodobně vyřešilo implementování tzv. `Loading baru`.
- Tlačítko `Clear Scene` neodstraní ze scény body. Je to z toho důvodu, že aplikace nenabízí uživateli body po jejich skrytí zase zobrazit. Opět stojí za zvážení aplikaci o tuto funkcionalitu rozšířit.
- Aplikace nenotifikuje uživatele o dokončení detekce kontaktů.
- Chybí možnost spustit všechny kroky algoritmu jedním stisknutím tlačítka.
- Sekce `Adjust Electrodes` je možná zbytečná, uživatel může naráz posunout všechny body konkrétní elektrody jednoduše najetím na jeden z bodů a podržením prostředního tlačítka myši body posunout.
- Rozšíření v jeho stávající podobně není připravené na to být součástí oficiálního repozitáře rozšíření 3D Sliceru. S trochou štěstí se toto v budoucnu změní. V současné době z toho ovšem pro uživatele vyplývá, že si rozšíření musí pro instalaci stáhnout z externího zdroje.

Kapitola 6
Závěr

6.0.1 Zhodnocení práce

V rámci této práce vzniklo rozšíření pro desktopovou aplikaci 3D Slicer, které implementuje algoritmus vyvinutý Janca a kol. [8], který slouží k identifikaci kontaktů SEEG elektrod po jejich implantaci do mozku v rámci léčby fokální epilepsie.
Cílem bylo modifikovat algoritmus tak, aby byl kompatibilní s platformou 3D Slicer a tím získal nejen uživatelské rozhraní ale zároveň se tak stal přístupným pro širokou veřejnost.
V rámci této práce jsem se musel naučit pracovat s platformou Slicer, naučit se tvořit rozšíření pro platformu Slicer a naučit se pracovat s knihovnami, kterých Slicer pro svou činnost využívá.
Výsledné rozšíření implementuje všechny požadavky specifikované v zadání práce a podle výsledků testování je schopné detekce kontaktů elektrod s přesností na 0.21 ± 0.10 milimetru.
Rozšíření implementuje grafické rozhraní schopné vizualizace některých mezivýsledků, stejně tak jako výsledných bodů.
Stylisticky sleduje grafické rozhraní styl používaný ve všech modulech aplikace Slicer, který není založen tolik na estetice, ale spíše na funkčnosti a přehlednosti.
Díky využití Slicer je rozšíření dostupné na všech operačních systémech, nicméně zatím není možné si aplikaci nainstalovat skrze instalátor rozšíření Sliceru, ale musí se nainstalovat externě.
Rozšíření obsahuje několik nedostatků, které jsou uvedeny v kapitole 5.2. Největším z nedostatků je, že aplikace v průběhu výpočtů není responzivní. To by se mohlo spravit implementací Loading barů, nebo alespoň vyskakovacím oknem upozorňujícím uživatele. Dalším z nedostatků jsou prvky uživatelského rozhraní, jako záložka `Adjust Electrodes` a `BRAINSFit modul`, které jsou pro průměrného uživatele zbytečné a rozšíření by pravděpodobně bylo uživatelsky pohodlnější, kdyby se tyto prvky jednoduše odstranili. Tyto a další nedostatky budou opraveny v druhé polovině roku 2024.
Rozšíření je společně s anglickým návodem na použití dostupné ke stažení zde
Tímto byly splněny všechny body zadání bakalářské práce.

Literatura

[1] John S Duncan, Josemir W Sander, Sanjay M Sisodiya a Matthew C Walker. Adult epilepsy. *Lancet*. 2006, 367 (9516), 1087–1100. DOI 10.1016/s0140-6736(06)68477--8.

[2] Philippe Ryvlin, J Helen Cross a Sylvain Rheims. Epilepsy surgery in children and adults. *Lancet neurology*. 2014, 13 (11), 1114–1126. DOI 10.1016/s1474-4422(14)70156-5.

[3] Chaturbhuj Rathore a Kurupath Radhakrishnan. Concept of epilepsy surgery and presurgical evaluation. *Epileptic disorders*. 2015, 17 (1), 19–31. DOI 10.1684/epd.2014.0720.

[4] M. Cossu, D. Fuschillo, F. Cardinale, L. Castana, S. Francione, L. Nobili a G. Lo Russo. Stereo-EEG-guided radio-frequency thermocoagulations of epileptogenic grey-matter nodular heterotopy. *Journal of neurology, neurosurgery and psychiatry*. 2013, 85 (6), 611–617. DOI 10.1136/jnnp-2013-305514.

[5] S. Medina Villalon, R. Paz, N. Roehri, S. Lagarde, F. Pizzo, B. Colombet, F. Bartolomei, R. Carron a C.-g. Bénar. EpiTools, A software suite for presurgical brain mapping in epilepsy: Intracerebral EEG. *Journal of neuroscience methods*. 2018, 303 7–15. DOI 10.1016/j.jneumeth.2018.03.018.

[6] Brian Ervin, Leonid Rozhkov, Jason Buroker, James L. Leach, Francesco T. Mangano, Hansel M. Greiner, Katherine D. Holland a Ravindra Arya. Fast Automated Stereo-EEG Electrode Contact identification and Labeling ensemble. *Stereotactic and functional neurosurgery*. 2021, 99 (5), 393–404. DOI 10.1159/000515090.

[7] Massimo Narizzano, Gabriele Arnulfo, Serena Ricci, Benedetta Toselli, Martin Tisdall, Andrea Canessa, Marco Massimo Fato a Francesco Cardinale. SEEG assistant: a 3DSlicer extension to support epilepsy surgery. *BMC bioinformatics*. 2017, 18 (1). DOI 10.1186/s12859-017-1545-8.

[8] Radek Janca, Martin Tomasek, Adam Kalina, Petr Marusic, Pavel Krsek a Robert Lesko. Automated identification of stereoelectroencephalography contacts and measurement of factors influencing accuracy of frame stereotaxy. *IEEE journal of biomedical and health informatics*. 2023, 27 (7), 3326–3336. DOI 10.1109/jbhi.2023.3271857.

[9] *3D Slicer image computing platform*. `https://www.slicer.org/`.

[10] Andriy Fedorov, Reinhard Beichel, Jayashree Kalpathy-Cramer, Julien Finet, Jean-Christophe Fillion-Robin, Sonia Pujol, Christian Bauer, Dominique Jennings, Fiona Fennessy, Milan Sonka, John Buatti, Stephen Aylward, James V. Miller, Steve Pieper a Ron Kikinis. 3D Slicer as an image computing platform for the Quantitative Imaging Network. *Magnetic resonance imaging*. 2012, 30 (9), 1323–1341. DOI 10.1016/j.mri.2012.05.001.

[11] *Module Overview — 3D Slicer documentation.* https://slicer.readthedocs.io/en/latest/developer_guide/module_overview.html.

[12] Jeff Novotny. *The Pros and Cons of Python Programming*. 2022. https://www.linode.com/docs/guides/pros-and-cons-of-python/.

[13] Rizel Scarlett. *Why Python keeps growing, explained - The GitHub Blog*. 2023. https://github.blog/2023-03-02-why-python-keeps-growing-explained/.

[14] https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html .

[15] *Coordinate systems — 3D Slicer documentation.* https://slicer.readthedocs.io/en/latest/user_guide/coordinate_systems.html.

[16] *Developing extension in 3D Slicer.* https://docs.google.com/presentation/d/1JXIfs0rAM7DwZAho57Jqz14MRn2BIMrjB17Uj_7Yztc/edit##slide=id.g41f90baec_028.

[17]

[18] *Modules:BRAINSFit - Slicer Wiki.* https://www.slicer.org/wiki/Modules:BRAINSFit.

[19] Ali Fawzi, Anusha Achuthan a Bahari Belaton. Brain Image Segmentation in recent Years: A Narrative review. *Brain sciences*. 2021, 11 (8), 1055. DOI 10.3390/brainsci11081055.

[20] Lassoan. *GitHub - lassoan/SlicerHDBrainExtraction: 3D Slicer extension for accessing HD-BET brain extraction tool for skull stripping in brain MRI images.* https://github.com/lassoan/SlicerHDBrainExtraction##hdbrainextraction.

[21] *Hounsfield units - scale of HU, CT numbers | Classifications, online calculators, and tables in radiology.* http://www.radclass.mudr.org/content/hounsfield-units-scale-hu-ct-numbers.

[22] *numpy.polyfit — NumPy v1.26 Manual.* https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html.

[23] Ransaka Ravihara. Gaussian Mixture Model: A Comprehensive Guide to Understanding and Implementing GMM from Scratch | Towards Data Science. 2023.