VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SEGMENTACE TOMOGRAFICKÝCH DAT V PROSTŘEDÍ 3D SLICER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

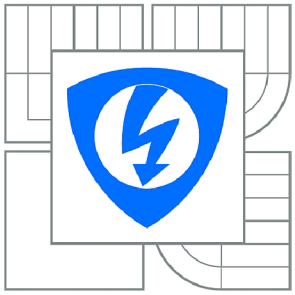
AUTOR PRÁCE
AUTHOR

Bc. ROBERT KORČUŠKA

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

SEGMENTACE TOMOGRAFICKÝCH DAT V PROSTŘEDÍ 3D SLICER

SEGMETATION OF TOMOGRAPHIC DATA IN 3D SLICER

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

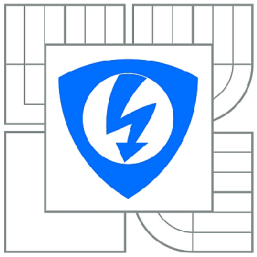
AUTOR PRÁCE — Bc. ROBERT KORČUŠKA

AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN MIKULKA, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Robert Korčuška	***ID:***	136543
Ročník:	2	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Segmentace tomografických dat v prostředí 3D Slicer

POKYNY PRO VYPRACOVÁNÍ:

Návrhněte vhodnou víceparametrickou metodu pro segmentaci tomografických dat a následně tuto metodu implementujte do prostředí 3D Slicer. Ověřte navrženou metodu segmentací vybrané skupiny pacientů s nádorovým onemocněním. Srovnejte výsledky navržené segmentační metody s běžnými jednoparametrickými metodami.

DOPORUČENÁ LITERATURA:

[1] Duda, R., O., Hart, P., E., Stork, D., G. Pattern Classification. John Wiley & Sons, 2001.
[2] Sonka, M., Hlavac, V., Boyle, R. Image Processing, Analysis, and Machine Vision.

Termín zadání:	9.2.2015	***Termín odevzdání:***	26.5.2015

Vedoucí práce: Ing. Jan Mikulka, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Táto práca obsahuje základný teoretický rozbor segmentácie obrazu pomocou techniky SVM, teóriu klasifikácie dát a popis programu 3D Slicer. Práca popisuje spracovanie medicínskych obrazov a demonštruje problematiku segmentácie týchto obrazov. Obsahuje návrh a implementáciu metódy SVM v programe 3D Slicer ako rozširovací modul programu. SVM metóda je porovnaná s jednoduchými segmentačnými metódami programu 3D Slicer. Experimentálne je overená kvalita segmentácie metódou SVM na reálnych subjektoch.

Kľúčové slová

SVM, klasifikácia dát, segmentácia obrazu, 3D Slicer, tomografické dáta

Abstract

This thesis contains basic theoretical information about SVM-based image segmentation and data classification. Basic information about 3D Slicer software are presented. Aspects of medical images segmentation are described. Workplan and implemetation of SVM method for MRI segmentation in 3D Slicer sofware as extension module is created. SVM method is compared with simple segmentation algorithms included in 3D Slicer. Quality of segmentation, based on SVM, tested on real subjects is experimentaly demonstrated.

Keywords

SVM, data classification, image segmentation, 3D Slicer, tomographic data

KORČUŠKA, R. *Segmentace tomografických dat v prostředí 3D Slicer*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 66 s. Vedoucí diplomové práce Ing. Jan Mikulka, Ph.D.

Prehlásenie

Prehlasujem, že som svoju diplomovú prácu na tému „Segmentace tomografických dat v prostředí 3D Slicer“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce. Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných alebo majetkových a som si plne vedomý následkov porušenia ustanovenia §11 a nasledujúcich autorského zákona č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka č. 40/2009 Sb.

V Brne dňa 20. mája 2015 .....

podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výskum popísaný v tejto diplomovej práci bol realizovaný v laboratóriách podporených z projektu SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno ....
(podpis autora)

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

Poďakovanie

Ďakujem vedúcemu diplomovej práce, pánu Ing. Janu Mikulkovi, Ph.D., za ochotu a dobrý prístup ku konzultáciám, účinnú metodickú, pedagogickú a odbornú pomoc a ďalšie cenné rady, pri spracovaní mojej diplomovej práce.

V Brne dňa 20. mája 2015 .....

podpis autora

OBSAH

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

CMake	Kompilačný program
CTK	Nástroj podporujúci spracovanie biomedicínskych obrazov
DCTMK	Nástroj na spoluprácu so štandardom DICOM
DICOM	Štandard prenosu dát v medicíne
GUI	grafické užívateľské rozhranie
ITK	Nástroj obsahujúci segmentačné a registračné nástroje
MIT	Massachusetts Institute of Technology
MR	Magnetická rezonancia
MRML	Dátový model popisujúci rozloženie scény v 3D Slicer
NA-MIC kit	Otvorená platforma obsahujúca 3D Slicer a mnoho ďalších doplnkov
NN	Neurónová sieť
PACS	Systém pre archiváciu a prenos obrazu
PR	Perfúzny MR obraz
Qt	multiplatformný framework
SEM	3D Slicer vykonávacie moduly
SVM	Support vector machines, metóda podporných vektorov
SVN	Verzovací software, alternatíva k programu Git
VTK	Vizualizačná sada nástrojov
$f(x)$	funkcia klasifikácie vektorov
K	jadrová funkcia
L_d	Lagrangeov duálny multiplikátor

L_p	Lagrangeov primárny multiplikátor
m	súčet najkratších vzdialeností vektorov k separačnej nadrovine
N	počet podporných vektorov
$\mathbf{P}$	set trénovacích dát SVM
R^D	D - rozmerný priestor
s_i	podporné vektory
w	kolmica k nadrovine
x_i	i-rozmerné vektory trénovacích dát
y_i	označenie i-tej triedy SVM
Φ	funkcia transformácie do viacrozmerného priestoru

ÚVOD

Lekárske vyšetrenia sa v priebehu posledných desiatok rokov výrazne zmenili. V dnešnej dobe narastá využívanie neinvazívnych techník medicínskeho zobrazenia. Tieto umožňujú lekárom analyzovať štruktúru tkanív a určiť diagnózu bez nutnosti invazívneho zákroku. Množstvo týchto vyšetrení, ale kladie vysoké nároky na lekára, ktorý musí každý obraz vyhodnotiť. V priebehu niektorých vyšetrení, ako napríklad počítačová tomografia obrazov, vzniká veľké množstvo a ich analýza je veľmi zdĺhavá. Do popredia preto vystupujú nástroje, ktoré uľahčujú prácu medika a napomáhajú pri správnom rozlišovaní tkanív a určení diagnózy. Špecializované algoritmy nemajú za cieľ úplne nahradiť človeka, pomáhajú mu napríklad zvýrazniť podozrivé regióny pri hľadaní nádorových ochorení, či vykonať predspracovanie obrazu, ako je odstránenie artefaktov a šumu.

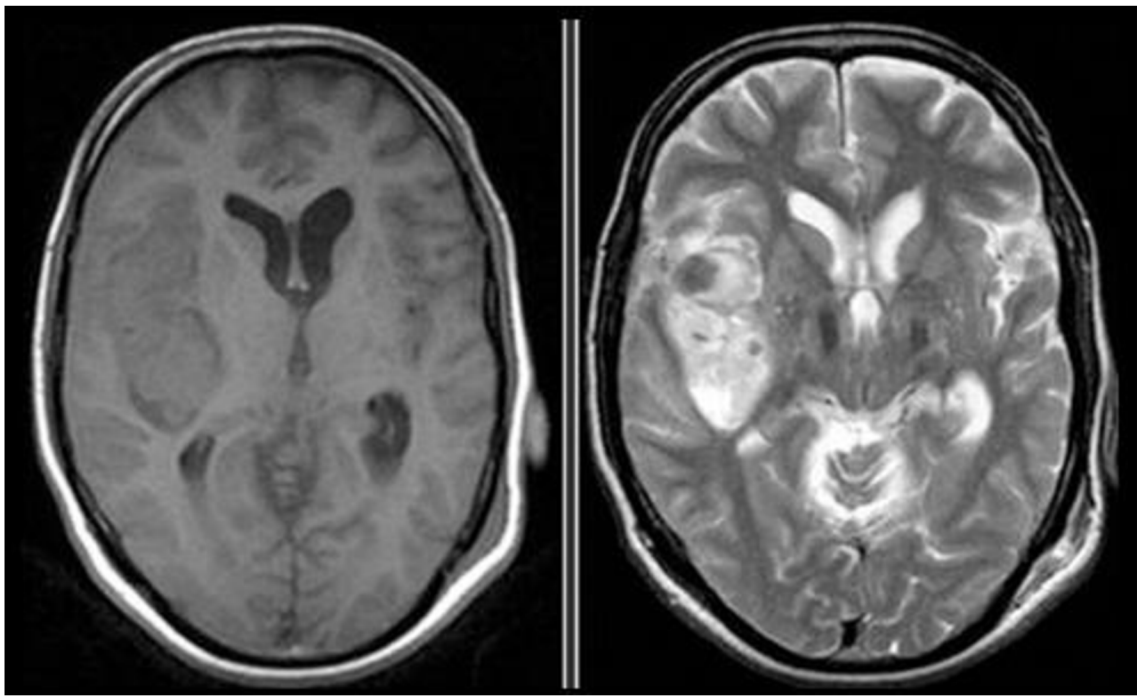
Táto práca je zameraná na nájdenie optimálneho spôsobu segmentácie tomografických dát v prostredí 3D Slicer. V tomto programe je následne vytvorené rozšírenie implementujúce túto segmentačnú metódu. Spracovanie obrazov MR je náročný proces. Získavanie informácie ktorá je užitočná pre medicínskeho špecialistu je zložitá z dôvodu, že MR obrazy sú zložené z viacerých typov obrazov snímaných za rôznych podmienok. Tieto obrazy spolu dávajú ucelený pohľad na štruktúru tkanív, ktorý môžeme využiť v segmentácií a nájdení regiónu záujmu. Statické obrazy zobrazujú štruktúrne zloženie mäkkých tkanív skúmaného subjektu. Dynamické obrazy snímané v čase, perfúzne obrazy, sú nositeľom informácie o aký typ tkaniva ide. Je to dôležité hlavne pri skúmaní nádorov a ich metastáz, kde z postupného prenikania kontrastnej látky je možné určiť typ tumoru a vhodne stanoviť diagnózu. Nevýhodou týchto obrazov je ich veľmi nízke rozlíšenie. Pre špecialistu je obtiažne s týmto obrazom pracovať aj z dôvodu, že tento typ obrazov neobsahuje hranice objektov. Diplomová práca skúma primárne mozgové tkanivá, statické štruktúrne obrazy sú vhodné pre toto spracovanie. Ich výhodou je dostatočné rozlíšenie ktoré umožňuje rozoznať objekty v jednotkách milimetrov. Nevýhodou je že tieto obrazy obsahujú každý iné potrebné informácie, a je potrebné využívať informačnú hodnotu z celku. Z toho dôvodu, je nutné použiť viac parametrickú metódu, ktorá vhodne využije údaje v obrazoch. Z výsledného segmentu je možné použiť masku v perfúznom obraze a získať tak informácie v skúmanom objekte oddelenom od ostatných štruktúr. Vhodnou a aktuálne často používanou je metóda podporných vektorov ktorej využitiu v medicínskom prostredí je táto práca venovaná. Program 3D Slicer bol zvolený z dôvodu jeho univerzálnosti, vzhľadom k podmienkam práce v experimentálnom prostredí a jeho otvorenej platforme.

Prvá kapitola je zameraná na popis problematiky 3D tomografických dát. Popisuje štruktúrne a perfúzne obrazy, ich dôležitosť v diagnostike. Ďalej obsahuje

základný popis nárokov kladených na software spracujúci medicínske dáta a obsahuje informácie o niektorých základných produktoch pre spracovanie týchto typov obrazov. Druhá kapitola popisuje základné segmentačné techniky a hľadá tak optimálnu variantu segmentačného algoritmu pre spracovanie medicínskych dát, ktoré sa vyznačujú svojou komplexitou. Tretia kapitola sa zaoberá sa metódou Support Vector Machine a obecne klasifikáciou dát, na ktorej je založená zvolená metóda segmentácie. Kapitola podrobne popisuje funkčnosť metódy. Štvrtá kapitola popisuje experimentálne využitie programu 3D Slicer. V úvode popisuje jeho platformu a architektúru. Popisuje jeho modularitu a možnosti tvorby jednoduchých rozšírení, na zvýšenie funkčnosti tohto softwaru. Ďalej popisuje implementáciu segmentačnej metódy SVM ako rozšírenie do tohto programu. Kapitola obsahuje aj základne prvky a metódy ktoré sú univerzálne použiteľné pri tvorbe ďalších rozšírení programu. V piatej kapitole sú prezentované výsledky navrhnutého riešenia. Príkladom je porovnanie s jednoduchými metódami a problematika potreby využitia viacerých parametrov vzhľadom k výslednej kvalite segmentácie. Poslednou časťou je testovanie úspešnosti segmentácie na viacerých testovacích subjektoch a celkové vyhodnotenie presnosti tejto metódy pri skúmaní MR obrazov mozgových tkanív.

1. 3D TOMOGRAFICKÉ DÁTA

Nukleárna magnetická rezonancia je neinvazívna metóda získavania snímok ľudského tela. MR zobrazenie využíva magnetické pole k získaniu signálu od rezonujúcich jadier. Najčastejšie sú nimi jadrá vodíku. Signály od protónov v molekulách vody u MR sú ovplyvňované okrem koncentrácie aj jej chemickými väzbami, pohybom molekúl a prietokom. Reprezentuje teda nielen fyzikálne, ale aj chemické vlastnosti snímanej scény. Pri získavaní obrazu dochádza k váhovaniu obrazu niekoľkými parametrami. Na základe relaxačnej doby T_1 vzniká obraz T_1-váhovaný, a na základe relaxačnej doby T_2 vznikne T_2-váhovaný obraz. T_1 a T_2 obrazy majú štruktúrny charakter. Na obrázku sú zobrazené obrazy typu T_1 a T_2. Na obraze typu T_2 môžeme vidieť v ľavej časti poškodenie mozgu s krvácaním, menej zreteľne na obraze typu T_1.



Obrázok 1.1: T_1 obraz (naľavo), obraz typu T_2 (napravo)

T_1 obraz mozgu v dostupnej databáze subjektov dosahuje rozlíšenie 320 × 320 voxelov v jednom reze, pričom je dostupných 209 rezov. Na tomto type obrazu je vidieť napríklad tumor tmavým odtieňom. Obraz typu T_2 dosahuje rozlíšenie 432 × 432 voxelov v jednom reze. Dostupných je 300 rezov. V tomto type obrazu je vidieť tumor svetlou farbou. Oba typy obrazov sa vzájomne odlišujú a pre klinického pracovníka majú inú výpovednú hodnotu. V tabuľke je vidieť základne rozdiely intenzít voxelov v zobrazovaní rôznych tkanivových štruktúr v T_1 a T_2 váhovaných obrazoch. Hodnoty sú zhodné s výstupnými informáciami z MR skeneru.

Typ tkaniva	MR- T_1 [-]	MR- T_2 [-]
Kostená štruktúra	547 ± 103	33,6 ± 2,8
Voda, mozgovomiešny tok	210 ± 61	72 ± 65
Vzduch	0	0
Tumor	1409 ± 202	1602 ± 252
Edém	429 ± 21	2636 ± 169

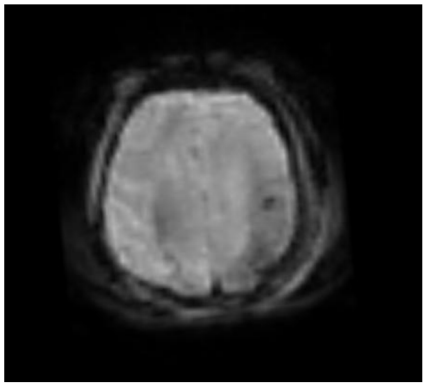
Tabuľka 1: Intenzity rôznych tkanív v obrazoch T_1 a T_2

Na určenie typu tumoru sa používajú perfúzne obrazy. Vyznačujú sa tým, že sú snímané v čase. Ide teda o viacnásobný kompletný sken. Zobrazuje postupné prenikanie kontrastnej látky do mozgových tkanív. Na základe odchýlok od štandardného rozptylu tekutiny špecialista dokáže určiť o aký typ tumoru ide. Nevýhodou tohto obrazu je že v jednom reze dosahuje rozlíšenie len 64 × 64 voxelov. Počet rezov v celom obraze je len 19.

Vlastnosti obrazu	MR- T_1	MR- T_2	MR - PR
Rozlíšenie	320 × 320 px	432 × 432 px	64 × 64 px
Počet rezov	209	300	19
Priestorové rozlíšenie pixelu	0,840 × 0,840 mm	0,578 × 0,578 mm	3,437 × 3,437 mm
Vzdialenosť medzi dvoma rezmi	0,8 mm	0,6 mm	3,6 mm

Tabuľka 2: Vlastnosti MR obrazov

V perfúznom obraze taktiež nie je vidieť hranice tumoru ani edému. Detail jedného rezu je možné vidieť na obrázku 1.2. Toto nízke rozlíšenie a nemožnosť nájdenia hraníc tak znemožňuje jednoznačné určenie typu a môže tak dôjsť k nesprávnemu určeniu diagnózy.



Obrázok 1.2: Perfúzne vážený obraz

Pretože pre diagnózu je potrebné analyzovať perfúzny obraz, je potrebné mať presne určené hranice tumoru prípadne edému, avšak hranice v perfúznom obrazne nie je vidieť. Tie je potrebné získať z štruktúrnych obrazov T_1 a T_2 kde hranice objektov sú zreteľnejšie. Po získaní segmentov rozlišujúcich tkanivá je potrebné tieto segmentované obrazy prevzorkovať na veľkosť perfúzneho obrazu. Vo výslednom obraze potom medicínsky špecialista nepotrebuje zložito premýšľať nad hranicami kde sa tumor ešte nachádza alebo nie.

1.1 FORMÁT TOMOGRAFICKÝCH DÁT

Digitálne obrazy z MR sú vytvorené v trojdimenzionálnom formáte. Obrazovou jednotkou 3D digitálneho obrazu je voxel. Voxel má plochu jedného pixelu a hĺbku rovnajúcu sa hrúbke rezu. Trojdimenzionálny obraz z MR si môžeme predstaviť ako množstvo dvojrozmerných obrazov zaznamenávajúcich objekt postupne za sebou. Hĺbka často nie je rovnaká ako ostatné rozmery voxelu, a preto sa využíva interpolácia na dopočítanie obrazu. Spolu s rôznymi typmi získavaných snímok získavame niekoľko typov 3D obrazov, na základe ktorých môžeme vykonávať analýzu a spracovanie.

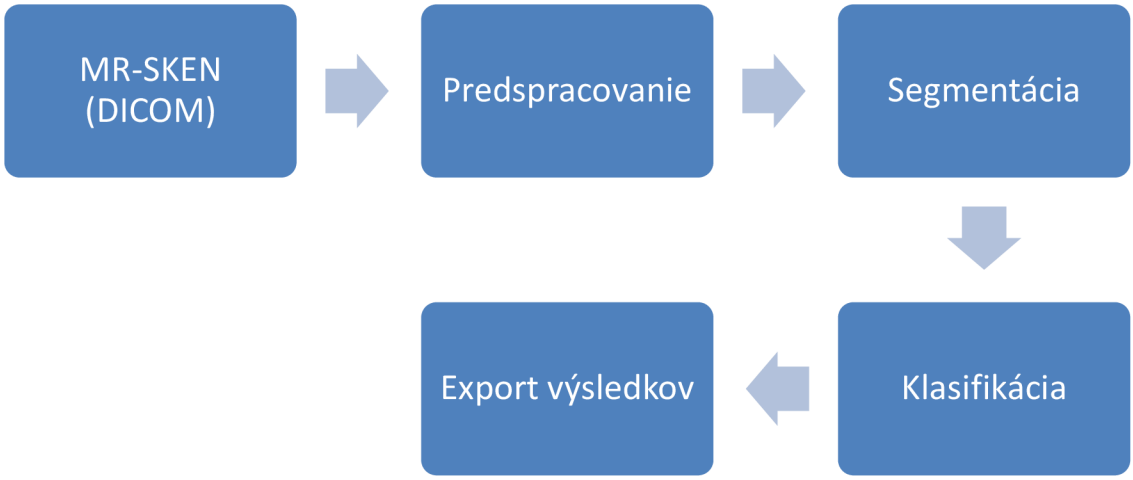
V minulosti pracovala väčšina medicínskych zobrazovacích techník v analógovej forme. Analýza dát však vyžadovala veľké množstvo času, ktoré musel stráviť expert nad vyhodnocovaním medicínskych snímok. S príchodom digitálnych prístrojov, ako je počítačová tomografia a magnetická rezonancia, sa možnosti analýzy a spracovania dát významne rozšírili. Zo začiatku pracovali aj tieto prístroje na analógovej báze. Postupom času vznikla požiadavka transportovať a spracovávať obrazy na viacerých pracoviskách. V medicíne je potrebné mať dostupné výsledky

v čo najkratšom čase a tým zachraňovať životy, preto vznikol systém, ktorý dokázal zaistiť pohodlný a všestranný prístup k diagnostickým obrazovým informáciám z digitálneho archívu v reálnom čase, systém pre archiváciu a prenos obrazu - PACS. Pri hľadaní a vytváraní počítačových systémov pre zobrazovanie obrazových dát vzniklo mnoho prenosových protokolov, väčšinou navzájom nekompatibilných. V roku 1983 vznikol štandard DICOM (Digital Imaging and Communications in Medicine), ktorý zaistil univerzálne kompatibilné rozhranie pre všetkých užívateľov, zahŕňajúci aj rozmanitý vyšetrovací a analyzačný software. DICOM formát zahŕňa v sebe okrem obrazových dát aj ďalšie rozširujúce informácie, napríklad informácie o parametroch vyšetrenia, číslo snímky, informácie o pacientovi, rôzne informácie potrebné pre prenos cez sieť a spracovanie obrazu. Protokol umožňuje sprístupniť obrazovú dokumentáciu v pôvodnej kvalite. Dnes je protokol DICOM schopný obsiahnuť väčšinu obrazov a signálov používaných v medicíne, ako aj iné výsledky prípadne lekárske správy, preto sa štandard DICOM rozšíril na použitie v celej nemocnici a stal sa univerzálnym štandardom pre prenos údajov v medicínskom prostredí.[1]

1.2 SOFTWARE NA SPRACOVANIE TOMOGRAFICKÝCH DÁT

Manuálna segmentácia 3D štruktúr je časovo veľmi náročná pre odborníkov, ktorý ju vykonávajú a vedie často k vytváraniu chýb a odchýlok vo výsledkoch. V 2D užívateľ kreslí priamo kontúry objektu, v 3D musí nakresliť okraje v každom reze objektom. Poloautomatické metódy vyžadujú manuálnu interakciu pri automatických krokoch. Interakcia spočíva v označovaní hraníc prípadne bodov, ktoré rozhodujú či sa jedná o hľadaný objekt alebo o pozadie. Poloautomatické metódy segmentácie umožňujú rozhodnúť špecialistom o správnosti výsledku prípadne opraviť parametre, k čomu potrebuje podporné nástroje pre tento účel.[2]

Analyzačných softwarov schopných spracovať a segmentovať 3D dáta z medicínskeho prostredia, spolupracovať s nemocničnými zariadeniami, prípadne vykonávať operácie ako segmentácia je mnoho. Poznáme ich od experimentálnych až po profesionálne korporátne aplikácie. S rastúcim množstvom dát získaných od pacientov a požiadaviek na diagnostiku týchto dát, sa tento software stal dôležitým smerom k zvýšeniu produktivity a vyhnutiu sa chybám. Na úplnú automatizáciu je ale potrebné dané systémy manuálne naučiť ako segmentovať rôzne štruktúry obrazov.



Obrázok 1.3: Proces spracovania MR dát z DICOM formátu

Na obrázku 1.3 je zobrazený proces spracovania obrazu. Po nasnímaní obrazu je nutné obraz predspracovať. Je to z toho dôvodu, aby sa zvýšila spoľahlivosť pri optickej inšpekcii. Používajú sa rôzne filtračné metódy na odstránenie šumu, tak aby sa zachovali dôležité dáta. V segmentácií sa obraz rozdelí na segmenty s podobnými atribútmi. V klasifikácií sa vyhodnotí predmet záujmu. Začína sa s viac diskriminačnými parametrami a postupne sa pridávajú menej diskriminačné parametre, pokým nie je možné klasifikačný výkon naďalej zlepšovať. Metódy, ktoré sa pri tomto procese používajú sú napríklad SVM, neurónové siete, metóda najbližších susedov (k-NN) a podobne. Po klasifikácií sa obraz s vyznačenými objektmi a ich klasifikáciou môže ďalej exportovať.[3]

3D Slicer je otvorená aplikácia používaná pri spracovávaní obrazu v medicíne ako klinický výskumný nástroj. 3D Slicer podporuje všestranné vizualizácie, ale taktiež poskytuje pokročilé funkcie ako je automatická segmentácia a registrácia rôznych aplikačných oblastí. 3D Slicer je zdarma a nie je viazaný na konkrétny hardware. Ako programovacia platforma, 3D Slicer zjednodušuje preklad a hodnotenie nových kvantitatívnych metód tým, že umožní výskumníkovi zamerať sa na implementáciu algoritmu a poskytuje abstrakciu pre bežné úkony dátovej komunikácie, vizualizácie a tvorbu užívateľského rozhrania. V porovnaní s ostatnými nástrojmi, ktoré poskytujú podobné funkcie, je 3D Slicer plne otvorený a je možné ho jednoducho rozšíriť. Program je navrhnutý tak, aby zjednodušil vývoj nových funkcií v podobe 3D Slicer rozšírení. [5]

Pre porovnanie software Matlab je v obecnom výskume tradičným nástrojom v oblasti vedeckých výpočtov. Mnoho výskumníkov používa Matlab pre vývoj prototypov a experimentovanie. Pretože Matlab nie je zameraný na lekárske aplikácie, chýba mu podpora pre rozhrania a zobrazenia v bežných klinických podmienkach. V dôsledku toho, nasadenie vyvinutých nástrojov pre použitie klinických výskumníkov vyžaduje preklad kódu do obecnejších jazykov, aby sa minimalizovali závislosti a zjednodušila integrácia, a preto nie je úplne vhodný na použitie v medicínskom prostredí.

2. SEGMENTAČNÉ METÓDY

Obecná definícia segmentácie hovorí, že je to proces delenia obrazu do častí, ktoré korešpondujú s konkrétnymi objektmi v obraze. Segmentácia je jeden z najdôležitejších krokov analýzy obrazu. Podľa prístupu môžeme rozdeliť segmentačné metódy na:

- metódy založené na detekcií hrán,
- metódy založene na detekcií regiónov,
- pokročilé metódy kombinujúce viacero spôsobov.

Osobitnou kapitolou sú medicínske obrazové dáta. Anatomická štruktúra tkanív nemusí byť dobre separovateľná, rovnako môže vznikať veľké množstvo šumu a nehomogenít. Na tieto obrazové dáta ale môžeme pozerať klasickou cestou spracovania obrazu. Dôležitým krokom je v tomto prípade segmentácia jednotlivých typov tkanív. Kvalitná segmentácia hrá kľúčovú rolu v nových metódach spracovania medicínskych dát.[7]

2.1 METÓDY ZALOŽENÉ NA DETEKCÍ HRAN

Hranami rozumieme body obrazu, v ktorých sa hodnota jasu prudko mení. Hranu môžeme chápať ako vlastnosť obrazového bodu, ktorá je reprezentovaná veľkosťou a smerom. Ideálna hrana môže byť skoková, ale v reálnych obrazoch je zmena jasu postupná. Proces detekcie hrán môžeme rozdeliť na tri fázy: filtráciu, diferenciáciu a detekciu. Pri filtrácií je odstránený šum, v diferenciácií sa zvýraznia oblasti v obraze a nakoniec sú detekované body kde je zmena intenzity najvýznamnejšia.[4]

Hranové detektory

Základné metódy detekcie hrán môžeme rozdeliť do dvoch hlavných skupín. Metódy využívajúce prvú deriváciu alebo druhú deriváciu. Pri použití prvej derivácie je výsledný hranový gradient porovnávaný s prahom, ktorý určuje, či sa jedná o hranu alebo nie. U metód druhej derivácie je výskyt hrany detekovaný, ak je priestorová zmena v polarite druhej derivácie dostatočne významná. Pretože MR obrazy sú zaťažené výrazným šumom je náročné zistiť ktorá zmena intenzity jasu patrí hrane a ktorá je daná šumom v obraze.[8]

Houghova transformácia

Je to segmentačná technika použiteľná vtedy, keď treba detekovať objekty so známym tvarom hranice. Houghova transformácia môže detekovať rovné čiary aj krivky (hranice objektu), ak sú známe ich analytické vyjadrenia. Je robustná pri identifikácii zakrytých a zašumených objektov.[8]

2.2 METÓDY ORIENTOVANÉ NA OBLASTI V OBRAZE

V predchádzajúcej časti boli popísané metódy na nájdenie hrán medzi dvoma oblasťami. Z týchto hrán je jednoduché vytvoriť oblasti a tiež je jednoduché vytvoriť hrany z oblastí. Avšak výsledky z použitia týchto metód nám nedávajú rovnaké obrazy, preto je často potrebná kombinácia metód založených na hranách a oblastiach. Metódy založené na oblastiach sa snažia v obraze nájsť oblasti, ktoré majú homogénne vlastnosti. Môže pri tom isť o jasovú intenzitu, farby, textúry alebo rôzne tvary. Homogenita je dôležitá vlastnosť oblasti a je použitá ako kritérium pri hľadaní oblastí. Jednoduchým kritériom homogenity je napríklad stredná hodnota jasovej intenzity. Segmentácia založená na hranách nám dáva korektné výsledky aj pri zašumených obrazoch, kde metódy založené na hranách zlyhávajú.

Medzi tieto metódy patria metódy spájania oblastí, rozdeľovania oblastí a ich vzájomná kombinácia, metóda rozdeľovania a spájania oblastí. Kombinácia delenia a spájania vedie k metóde, ktorá má výhody oboch prístupov. Techniky delenia a spájania obyčajne požívajú pyramídové obrazové reprezentácie. Pretože je k dispozícii aj delenie aj spájanie, počiatočný obrázok nemusí spĺňať ani podmienku homogenity oblasti, ani maximálnosti homogénnej oblasti.[8]

Metóda prahovania

Metóda prahovania je najstaršou segmentačnou metódou, ktorá je využívaná v jednouchých prípadoch aj dnes. Vzhľadom k jej výpočtovej nenáročnosti je najrýchlejšou segmentačnou metódou. Princíp vychádza zo skutočnosti, že mnoho objektov z oblasti obrazu je charakterizované konštantnou odrazivosťou či pohltivosťou svojho povrchu. Je teda možné využiť určenú jasovú konštantu, prah, na oddelenie objektov od pozadia. V MR obrazoch sa väčšinou nepoužíva pretože obrazy sú zaťažené šumom a objekty nemajú konštantnú intenzitu obrazu. Využíva sa ako súčasť sofistikovanejších metód.

Adaptívne prahovanie

Adaptívne prahovanie používame v prípadoch, keď vstupný obraz nie je dostatočne osvetlený alebo objekty v obraze vrhajú tieň. Pri takýchto objektoch pevne zvolená prahová hodnota nám dáva veľmi zlé výsledky. Z toho dôvodu budeme počítať prahovú hodnotu pre každý obrazový bod zvlášť. Výslednú prahovú hodnotu teda dostaneme na základe určitej lokálnej vlastnosti obrazového bodu a na základe jasovej intenzity daného obrazového bodu. Ak hovoríme o lokálnej vlastnosti rozumieme tým malé okolie obrazového bodu, pre ktorý počítame prahovú hodnotu. Okolie musíme zvoliť tak, aby obsahovalo dostatočne veľký počet bodov objektu aj pozadia. Výslednú prahovú hodnotu z tohto okolia vypočítame napríklad priemerovaním jasových intenzít,

mediánom alebo pomocou strednej hodnoty minimálnej a maximálnej jasovej intenzity.[8]

Metóda rozvodia

Metódu rozvodia je možné zaradiť medzi metódy orientované na oblasti v obraze. Táto morfologická metóda segmentácie je postavená na myšlienke pochádzajúcej z geografie. Obraz je chápaný ako terén alebo topografický reliéf, ktorý je postupne zaplavovaný vodou. Táto metóda sa často používa s medicínskych aplikáciách, hlavne v segmentácii mozgu. Jej výhodou je že je to osvedčená a efektívna metóda. Nevýhoda tejto metódy je v prípade veľkého zašumenia obrazu. Metóda produkuje veľký počet regiónov ktoré je nutné ďalej spracovať.

2.3 POKROČILÉ METÓDY

Aktívne kontúry

Aktívne kontúry, tiež nazývané ako snakes, zaznamenali počas svojho vývoja dômyselný prístup k extrakcii kontúr (obrysov) objektov a rozpoznávania obrazov. Obhajujú tvrdenie, že prítomnosť hrany nezávisí len od gradientu v určitom bode, ale aj na priestorovom rozložení. Aktívne kontúry včleňujú tento všeobecný názor hranovej detekcie pomocou ocenenia spojitosti a zakrivenia, kombinovaných s lokálnou silou hrany. Hlavnou výhodou oproti ostatným technikám je integrácia obrazových dát a počiatočného odhadu, požadovaných vlastností kontúry a obmedzení založených na znalostiach v jednotlivom extrakčnom procese. Technika hľadá krivku ohraničujúcu oblasť v obraze, ktorá spĺňa hľadané minimum energetickej funkcie modelu danej metódy. Kontúra má dve energetické funkcie, ktoré sú s ňou združené, vnútornú a vonkajšiu. Vnútorná miera energie požaduje vlastnosti formy kontúry také, ako hladkosť a spojitosť. Vonkajšia energia je odvodená od výberu typu obrázku, miera požaduje funkcie také, ako hrany, obrysy, regióny a najnovšie textúry. Vnútorná energia ako prioritná informácia, upravuje vonkajšiu energiu - druhotnú informáciu. Technika vylepšuje odhad počiatočnej kontúry pomocou techniky minimalizácie energie. Výhodou tejto metódy je možnosť prispôsobenia vlastností tak aby nebola citlivá na šum a prerušené hrany. Takto je možné segmentovať obraz aj bez predchádzajúceho predspracovania obrazu.[9]

Neurónové siete

Väčšina metód segmentácie obrazu je založená na znalostiach a skúsenostiach, ako a podľa čoho by mala prebiehať. Opakom je segmentácia obrazu neurónovými sieťami, ktorá nie je založená na podobných meta pravidlách. Trénovanie neurónovej siete čisto

orientovanej na dáta prebieha podľa princípu "učenia na príkladoch". Na druhej strane ležia algoritmy analyzujúce dáta vzhľadom k zadanej množine pravidiel a príznakov.

V zásade existujú dve stratégie trénovania umelej NN. Prvý prístup hľadá charakteristické vlastnosti vstupných dát (obvykle príznakové vektory) a klasifikuje ich do tried bez akejkoľvek ďalšej interpretácie. Tento prístup nazývame učenie bez učiteľa. Druhý prístup, tzv. trénovanie s učiteľom, vyžaduje ručne segmentované trénovacie dáta. Vstupom učiaceho algoritmu sú nielen príznakové vektory, ale aj funkcie, ktorá každému vstupnému vektoru priradzuje určitý segment obrazu.[3]

SVM

SVM sú v svojej podstate veľmi podobné umelým neurónovým sieťam. Hlavnými výhodami SVM oproti neurónovým sieťam je fakt, že pri použití SVM nám postačí spočítať jedno globálne minimum štruktúrneho rizika, zatiaľ čo pri neurónových sieťach počítame empirické riziko, ktoré sa nachádza (spravidla) v niekoľkých lokálnych minimách. SVM je dobre využiteľný vo viacrozmerných priestoroch.[10]

3. KLASIFIKÁCIA DÁT

Pod klasifikáciou dát rozumieme rozdelenie skupiny alebo množiny objektov, javov či procesov na konečný počet čiastkových skupín, podmnožín, v ktorých majú všetky jednotky dostatočne podobné spoločné vlastnosti. Klasifikačnými kritériami označujeme vlastnosti podľa ktorých, je možné klasifikáciu zadať alebo vykonávať. Klasifikačnú triedu tvoria objekty, ktoré majú podobné vlastnosti. Každá klasifikácia musí byť úplná. Každý predmet musí patriť do nejakej triedy a nemôže byť súčasne v dvoch či viac triedach.

Klasifikáciu vykonávame pomocou klasifikátora, čo je algoritmus so vstupom, zodpovedajúcemu charakteru dát, popisujúcich analyzovaný objekt a jedným diskrétnym výstupom. Hodnota výstupu je identifikátorom klasifikačnej triedy, do ktorej klasifikátor zaradí vstupné dáta podľa vzťahu

$$y_i = d(X). \tag{3.1}$$

Teda platí kde $d(X)$ je funkcia argumentu X predstavujúceho reprezentáciu vstupných dát, ktorú nazývame rozhodovacie pravidlo klasifikátora a ω_r, $r = 1, \ldots, R$ je identifikátor klasifikačnej triedy. Rozhodovacie pravidlo sa stanoví v tzv. učebnej fáze. Premennú X, formálne popisujúcu klasifikovaný objekt, nazývame obrazom.

Mnoho metód strojového učenia sa vyznačuje jednoduchými a efektívnymi algoritmami učenia, napríklad jednovrstvové umelé neurónové siete. Z hľadiska riešenia obecnej úlohy nájsť hranice, ktoré oddeľujú určité triedy vo vstupnom priestore, sú veľmi silne obmedzené schopnosťou naučiť sa len lineárne oddeľovače ako napr. priamky, roviny, nadroviny.

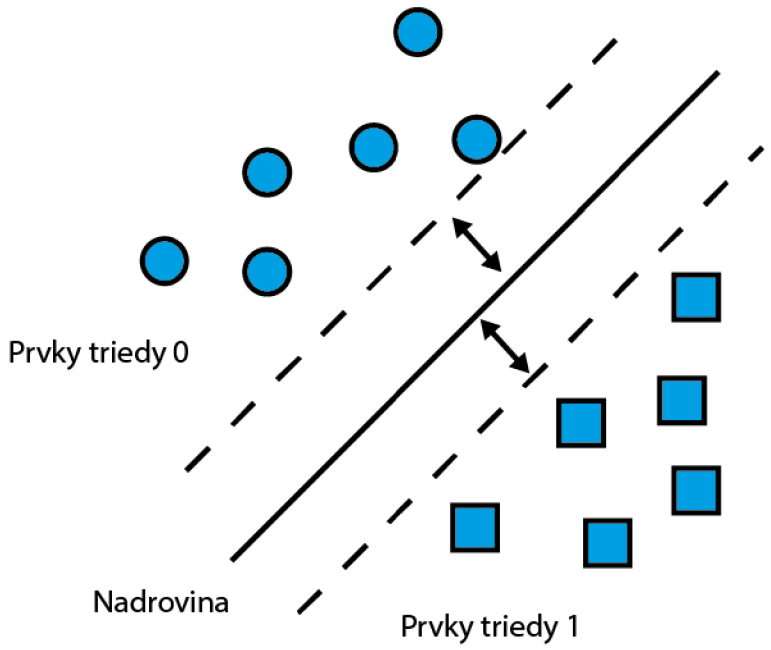
Na druhej strane existujú metódy, ako mnohovrstvové umelé neurónové siete, ktoré sú schopné reprezentovať obecné nelineárne funkcie. Ich nevýhodou však je často veľmi zložité učenie, pretože existuje riziko uviaznutia v lokálnom minime chybovej funkcie a navyše je učenie silne komplikované hľadaním vysokého počtu váh v mnohodimenzionálnom priestore.

3.1 SUPPORT VECTOR MACHINE

K alternatívnym metódam patria mechanizmy podporných vektorov SVM, ktoré tvoria určitú kategóriu tzv. kernel machines. Tieto metódy využívajú výhody poskytované efektívnymi algoritmami pre nájdenie lineárnej hranice a zároveň sú schopné reprezentovať vysoko zložité nelineárne funkcie. Jedným zo základných princípov je prevod daného pôvodného vstupného priestoru do iného, viacdimenzionálneho, kde je možné od seba oddeliť triedy lineárne.[11]

Neurónové siete a SVM sú učiace techniky, ktoré využívajú učenie na modeli alebo na vzore založenom na cvičných dátach, najviac tento model využívajú na predikciu alebo klasifikáciu dát. Aktívny vývoj neurónových sietí začal v roku 1970 a vývoj SVM datujeme do roku 1980. V súčasnosti je využívanie oboch techník učenia značne rozšírené. SVM vykazujú vysoký výkon na rozbore rôznych problémov, ktoré riešia aj neurónové siete. V súčasnosti sa očakáva rozširovanie využívania SVM, aj keď neurónové siete sú už do značnej miery rozšírené a využívané v softwarovom priemysle. Zatiaľ čo neurónové siete sa snažia riešiť špecifický problém a optimalizovať výkon, SVM využívajú systematický prístup a môžu byť jednoducho a priamo aplikované na rôzne problémové oblasti. Algoritmus využívajúci jadrové funkcie je v praxi využívaný napríklad pre rozoznávanie rukou písaných čísel, alebo na úlohy s veľmi vysokým počtom atribútov, úspešne funguje tiež ako filter napríklad textových dokumentov, ktoré sa často vyznačujú desiatkami tisíc atribútov (rôznych slov). [11]

SVM je typický algoritmus strojového učenia, hľadajúci nadrovinu, ktorá v priestore príznakov optimálne rozdeľuje cvičné dáta. Algoritmus slúži k lineárnej separácii dát, aj takých, ktorých separácia nie je možná. Požiadavkou pri hľadaní nadroviny je vzdialenosť medzi nadrovinou a najbližším prvkom jednotlivých tried (obrázok 3.1). Cieľom SVM je nájdenie iba jedného optimálneho lineárneho oddeľovača. Optimálny lineárny oddeľovač poskytuje čo najširšie pásmo medzi ním a pozitívnymi príkladmi na jednej strane a negatívnymi na druhej.



Obrázok 3.1.: Princíp klasifikácie

V okolí nadroviny je po oboch stranách čo najširší pruh, v ktorom sa nenachádza žiaden iný bod. K popisu tejto nadroviny slúžia najbližšie body, ktorých býva veľmi

málo. Tieto body sa nazývajú „podporné vektory“. Svoje meno získala metóda podľa týchto vektorov. Metóda SVM je binárna, čo znamená, že dáta rozdeľuje do dvoch tried.

Dôležitou súčasťou metódy SVM je jadrová transformácia. Táto umožňuje previesť pôvodne neseparovateľnú úlohu na úlohu separovateľnú, na ktorú je možné aplikovať ďalší optimalizačný algoritmus pre nájdenie rozdeľujúcej nadroviny. Výhoda tejto metódy je tá, že sa transformácia dá použiť aj na rôzne typy predmetu, napríklad rozhodovacie stromy a grafy.[11]

3.2 ROZDELENIE SUPPORT VECTOR MACHINE

SVM zahŕňa veľké množstvo algoritmov, ktoré môžeme rozdeliť do dvoch skupín podľa spracovania dát:

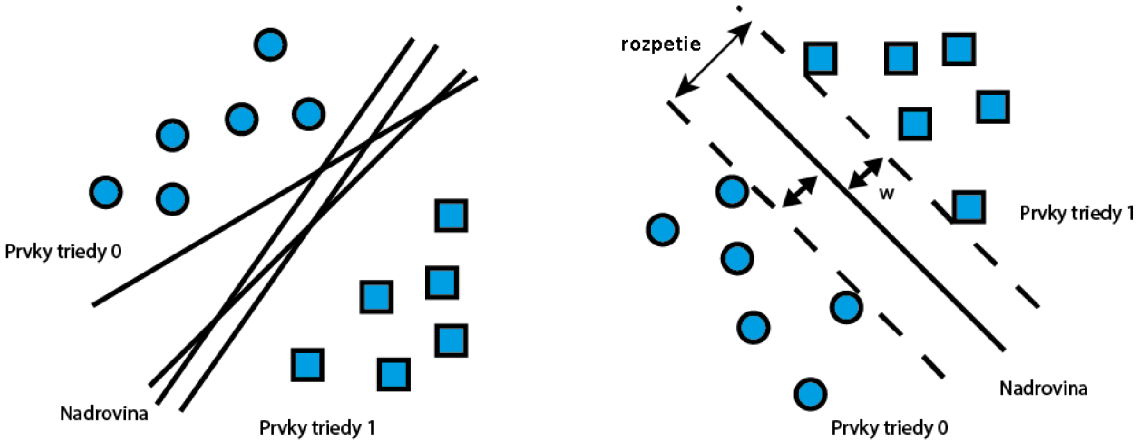
- lineárne separovateľné dáta,
- lineárne neseparovateľné dáta.

Lineárny model umožňuje špecifikovať a analyzovať vzájomné vzťahy viacerých premenných. Tento model sa rieši metódou najmenších štvorcov. V dnešnej dobe je dostupné veľké množstvo algoritmov na zostavenie konkrétnych modelov, ktoré zahŕňajú v sebe lineárne autoregresívne modely, modely kĺzavých súčtov, miešané a miešané integrované modely. Keďže prevažná časť reálnych procesov, ktoré sa snažíme modelovať, je nelineárneho charakteru a lineárne modely sú na ich modelovanie nepostačujúce, je potrebné použiť nelineárne modely.

Nelineárne modely majú na druhej strane vysokú výpočtovú náročnosť, takže donedávna nebolo možné tieto metódy používať v reálnom čase. S rýchlym vývojom výpočtovej techniky, najmä zvyšovaním výkonnosti počítačov, sa v tejto oblasti najviac presadili umelé neurónové siete.

SVM pre lineárne separovateľný priestor

Lineárne separovateľné cvičné dáta môžu mať nekonečný počet riešení a práve metóda SVM hľadá riešenie s čo najväčšou vzdialenosťou medzi dvoma typmi dát v cvičnej množine. Ide teda o hľadanie maximálnej vzdialenosti, ktorá rozdeľuje nadroviny klasifikátorov od bodov z cvičnej množiny. Táto vzdialenosť sa nazýva rozpätie. Pre lepšie vysvetlenie je toto separovanie pomocou maximálnej vzdialenosti naznačené na obrázku 3.2.[12]



Obrázok 3.2: Nekonečný počet riešení pri lineárne separovateľnom prípade (naľavo) a princípoch separácie pomocou maximálneho rozpätia (napravo)

Určovanie lineárne separovateľného priestoru

Označíme si skupinu cvičných dát:

$$\mathbf{P} = \left\{(x_i, y_i), x_i \in R^d, y_i \in \{-1,1\}\right\}_{i=1}^{l}, \tag{3.2}$$

kde každý bod x_i je d – rozmerný reálny vektor, y_i nadobúda hodnoty [-1,1] a označuje triedu do ktorej vektory x_i patria, y_i, taktiež sa označuje ako label tried. Máme konkrétnu separačnú nadrovinu, ktorá oddeľuje pozitívne a negatívne príklady (triedy). Takáto separačná nadrovina sa dá napísať ako set bodov x, ktoré vyhovujú vzorcu:

$$w \,.\, x + b = 0, \tag{3.3}$$

kde w je kolmica k nadrovine (obr. 3.2), $|b|/||w||$ je kolmá vzdialenosť od nadroviny k začiatku a $||w||$ je euklidovská forma w.[13] Nech d_+ (d_-) je najkratšia vzdialenosť od separačnej nadroviny k najbližšiemu pozitívnemu (negatívnemu) príkladu, potom môžeme definovať rozpätie ako ich súčet:

$$m = d_+ + d_-. \tag{3.4}$$

Keďže ide o lineárne separovateľný príklad, algoritmus SVM bude hľadať separačnú nadrovinu s najväčšou vzdialenosťou m. Keď sa podarí získať takúto nadrovinu s maximálnym rozpätím, všetky cvičné dáta budú spĺňať nasledovné nerovnice.[14]

$$x_i \cdot w + b \geq +1 \; pre \; y_i = +1, \tag{3.5}$$

$$x_i \cdot w + b \leq -1 \; pre \; y_i = -1, \tag{3.6}$$

tieto sa dajú spojiť do jednej nerovnice:

$$y_i(x_i \cdot w + b) - 1 \geq 0 \; pre \; \forall i. \tag{3.7}$$

Vektory, pre ktoré platí rovnosť 3.5 ležia na nadrovine $\mathbf{H_1}$, ktorá je daná rovnicou:

$$x_i \cdot w + b = 1, \tag{3.8}$$

podobne vektory pre ktoré platí rovnosť 3.6 sú umiestnené na nadrovine $\mathbf{H_2}$:

$$x_i \cdot w + b = -1, \tag{3.9}$$

kde w je opäť kolmica k tejto nadrovine a kolmá vzdialenosť od začiatku pre $\mathbf{H_1}$ bude $|1-b|/\|w\|$ a pre $\mathbf{H_2}$ $|-1-b|/\|w\|$. Vzdialenosť medzi $\mathbf{H_1}$ a $\mathbf{H_2}$ je teda $2/\|w\|$. Pre jednoznačne separovateľný set dát sa nebudú žiadne vektory nachádzať medzi nadrovinami $\mathbf{H_1}$ a $\mathbf{H_2}$ a teda maximálne rozpätie môžeme nájsť minimalizovaním $\|w\|$ pri dodržaní podmienky vo vzťahu 3.7.[14]

Po zadaní uvedených kritérií môžeme očakávať, že riešenie pre typický dvojtriedový, lineárne separovateľný príklad bude mať formu ako na obr. 3.2. Cvičné vektory, pre ktoré platí vzťah 3.7, to znamená, že ktoré skončia ležiace na jednej z nadrovín $\mathbf{H_1}$, $\mathbf{H_2}$ a ktorých odstránenie by zmenilo práve nájdené riešenie, sa nazývajú podporné vektory. Na obr. 3.2 sú tieto podporné vektory dotýkajúce sa body priamok. Keďže nám ostal problém minimalizácie $||w||^2$ vzhľadom na podmienku nerovnice 3.5, ktorý je náročný na výpočet, použijeme Lagrangeovu formuláciu tohto problému. Zavedieme kladné Lagrangeove multiplikátory (násobitele) jeden pre každé obmedzenie v nerovnici 3.6. Funkcia pre optimalizačný problém je potom daná nasledovne: [12]

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i \, y_i \, (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i \, . \tag{3.10}$$

Tento problém je možné riešiť štandardnými algoritmami kvadratického programovania. Keďže nám ale ide o minimalizáciu L_p, ktorá sa nazýva primárna formulácia a jej gradient s ohľadom na w a b by mal byť rovný nule, toto nám dáva systém rovníc:

$$w = \sum_{i=1}^{l} \alpha_i \, y_i \, x_i \tag{3.11}$$

$$\sum_{i=1}^{l} \alpha_i \, y_i = 0, \tag{3.12}$$

ak nazveme kvadratický optimalizačný problém vo vzťahu 3.10 primárny problém, môžeme zadefinovať duálny problém L_d rovnakého typu ako L_p tak, že Lagrangeove multiplikátory L_p sú súčasťou riešenia L_d a naopak. V mnohých prípadoch je riešenie duálneho problému jednoduchšie pre jeho štruktúru .[13]

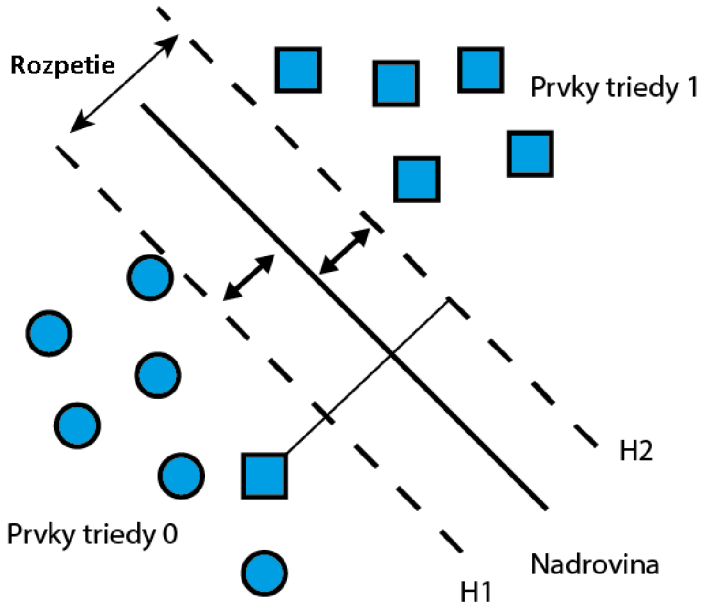
Po dosadení vzťahov 3.11 a 3.12 do 3.10 dostávame:

$$L_d = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j \quad (3.13)$$

SVM trénovanie (pre lineárny, separovateľný prípad) je teda možné vyjadriť ako snahu maximalizovať L_d s ohľadom na v_a, podliehajúci podmienkam vo vzťahu 3.12 a kladnosti α_i s riešením daným vzťahom 3.11. Musíme si uvedomiť, že pre každý trénovací vektor existuje aj Lagrangeov multiplikátor α_i. Vektory x_i, pre ktoré sú v riešení (dané vzťahom 3.11) Lagrangeove multiplikátory $\alpha_i > 0$ sú nami hľadané podporné vektory a ležia práve na jednej z nadrovín $\mathbf{H_1}$, $\mathbf{H_2}$. Pre SVM sú podporné vektory najdôležitejšou zložkou setu cvičných dát, pretože aj keby sme premiestnili všetky ostatné vektory tak, aby neprekročili nadroviny $\mathbf{H_1}$, $\mathbf{H_2}$, prípadne aby na nich neležali a opakovali by sme cvičenie, rovnaká separačná nadrovina by bola nájdená.[14]

SVM pre lineárne neseparovateľný priestor

Algoritmus určovania lineárnych SVM uvedený v predchádzajúcej kapitole bude platiť len pokiaľ budú dáta separovateľné, ako náhle budú dáta viac podobné a teda viac premiešané v priestore, vyskytnú sa neseparovateľné prípady – obrázok 3.3. Keby takýto prípad nastal, viedlo by to k svojvoľnému narastaniu cieľovej funkcie a teda algoritmus použitý na separovateľný prípad by skončil chybou. Spôsob počítania optimálnej separačnej nadroviny pre neseparovateľný prípad vyvinuli v roku 1995 V. Vapnik a C. Cortes [13].



Obrázok 3.3: Lineárne neseparovateľný prípad

Určenie lineárne neseparovateľného priestoru

Ak nastane prípad, že dáta jednej triedy sa nedajú jednoznačne oddeliť od druhej triedy, pričom počet vektorov, ktoré spôsobujú takúto chybu je malý v porovnaní s celkovým počtom vektorov, budeme musieť zaviesť kladné premenné ξi, i = 1,2,...,l, ktoré reprezentujú cvičebné chyby.

$$y_i(x_i \cdot w + b) - 1 \geq -\xi, \ \xi \geq 0. \tag{3.14}$$

Pre získanie optimálnej separačnej nadroviny, pri ktorej počet dát ktoré nemajú maximálne rozpätie je minimálny, budeme musieť minimalizovať cieľovú funkciu s ohľadom na vzťah 3.13. Cieľová funkcia je teda zväčšená o funkciu, ktorá penalizuje nenulové premenné ξ. Optimalizácia teda spočíva v rovnováhe medzi najväčším rozpätím pri najmenšom možnom počte chýb. Podmienka vo vzťahu 3.14 spolu s úlohou minimalizovania $||w||$ môže byť riešená ako v separovateľnom prípade. Riešenie optimálnej separačnej nadroviny je potom dané vzťahom:[2]

$$w = \sum_i^N = 1. \alpha_i y_i x_i \,, \tag{3.15}$$

kde N_s je počet podporných vektorov [11].

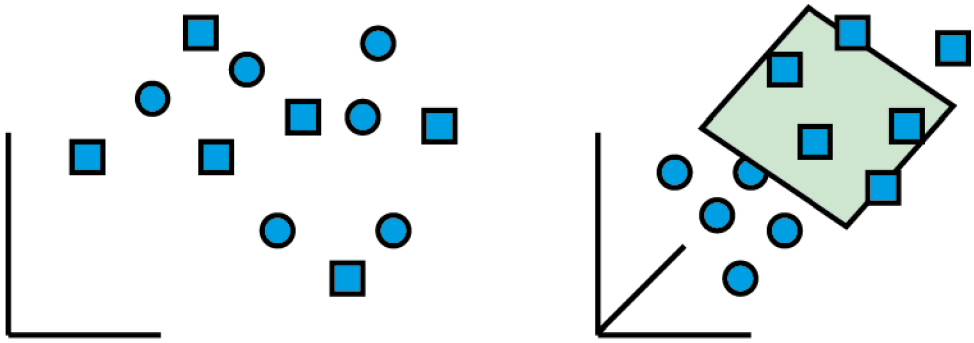
Hodnotenie neznámych vektorov pre lineárne separovateľné prípady môžeme vykonať pomocou funkcie:

$$f(x) = \sum_{i=1}^{l} \alpha_i \ y_i x_i x + b, \tag{3.16}$$

kde znamienko $f(x)$ udáva príslušnosť vektorov k jednej z dvoch tried.

Nelineárne SVM

Napriek definovaniu miery tolerancie v okolí separačnej nadroviny pre neseparovateľné prípady tento postup nedokáže riešiť mnoho reálnych problémov (obr. 3.4). Sila metódy SVM je práve v nelineárnej klasifikácií, pretože s pomocou jadrových metód môžeme vziať ľubovoľnú množinu cvičebných dát **P**, ktorá nie je lineárne separovateľná a transformovať ju do viacrozmerného priestoru. Vo viacrozmernom priestore je veľká pravdepodobnosť, že nájdeme lineárnu separačnú nadrovinu, ako je to znázornené na obr. 3.4.[12]



Obrázok. 3.4: Lineárne neseparovateľné dáta (naľavo) a ich lineárna separácia po transformácií do viacrozmerného priestoru (napravo)

Vo všetkých zápisoch doterajších riešení sa vyskytujú vstupné dáta len vo forme skalárneho súčinu, takže našou úlohou bude ich previesť do viacrozmerného (až nekonečného) priestoru $\mathbf{R}^D$ pomocou funkcie [14]:

$$\Phi: \mathbf{R}^d \rightarrow \mathbf{R}^D, \tag{3.17}$$

kde D je rozmernosť nového priestoru **R**. V tomto novom viacrozmernom priestore môžeme tiež vypočítať optimálnu separačnú nadrovinu s tým rozdielom, že skalárny súčin bude nahradený $\Phi(x_i).\Phi(x_j)$, čo predstavuje transformovanie vstupných vektorov do viacrozmerného priestoru a následné počítanie ich skalárneho súčinu. Ak by sme teraz dokázali zadefinovať jadrovú funkciu K, ktorá by dokázala tento skalárny súčin zo vstupov x_i a x_j vypočítať v trénovacom algoritme, nemuseli by sme explicitne vykonávať transformáciu vstupov do viacrozmerného priestoru Φ.

Pre jadro K v tomto prípade teda platí:

$$K(x_i x_j) = \Phi(x_i)\Phi(x_j). \tag{3.18}$$

Všetky predchádzajúce úvahy platia, keďže sa stále jedná o lineárnu separáciu dát, len v inom priestore. Klasifikácia neznámeho vektoru sa dá vypočítať nasledovne [11]:

$$f(x) = \sum_{i=1}^{l} a_i y_i \Phi(s_i)\Phi(x) + b, \tag{3.19}$$

po dosadení jadra zo vzťahu 3.16 dostaneme:

$$f(x) = \sum_{i=1}^{l} a_i y_i K(s_i, x) + b, \tag{3.20}$$

znamienko $f(x)$ opäť reprezentuje príslušnosť vektorov k jednej z dvoch tried a si sú podporné vektory [14].

Jadrové funkcie používané v nelineárnych SVM klasifikátoroch

Keďže problematika jadier používaných v SVM je rozsiahla a neustále sa vyvíja, dala by sa jej venovať samostatná práca. V tejto časti budú spomenuté len základne informácie o najpoužívanejších jadrách a ich matematické princípy. Detailnejšie informácie ohľadom jadier sa dajú nájsť v literatúre. Všetky funkcie, ktoré spĺňajú tzv. Mercerovu podmienku môžu byť považované za jadrové funkcie [12].

Najbežnejšie sa používajú tri jadrové funkcie:

• Gaussova (RBF),

• polynomiálna,

• sigmoidálna.

Gaussovské jadro má formuláciu:

$$K(x, y) = e^{(-\gamma\, ||x-y||^2)}, \tag{3.21}$$

kde γ je pozitívny parameter na regulovanie rádiusu [4].

Gaussovské jadro je asi najpoužívanejší zo všetkých vďaka svojej univerzálnosti a dobrej úspešnosti v mnohých oblastiach.

Polynomiálne jadro:

$$K(x, y) = (x^T y + 1)^d, \tag{3.22}$$

kde d je prirodzené číslo a udáva stupeň tohto jadra. Ak $d = 1$ toto jadro sa bude správať ako lineárne jadro + 1.

Sigmoidálne jadro:

$$K(x, y) = tanh(\gamma xy + c), \tag{3.23}$$

pričom γ a c sú parametre tohto jadra a možno ich charakterizovať [1]:

pre $\gamma > 0$:

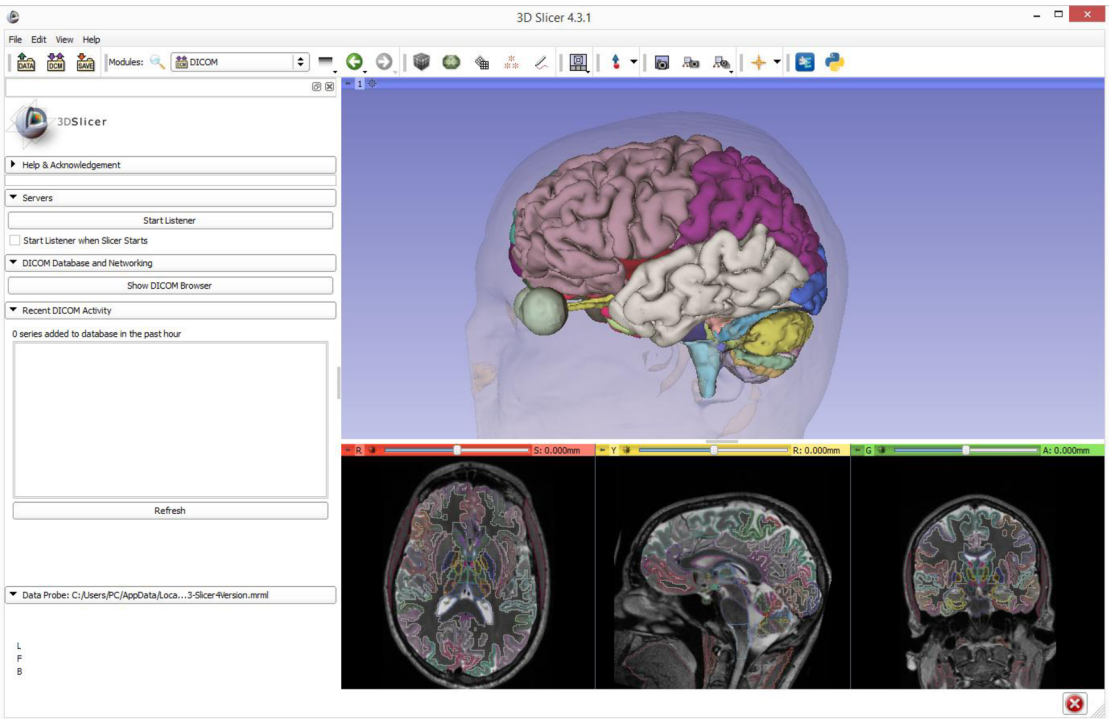
- γ bude parameter upravujúci hodnoty vstupných dát,
- c ako parameter, ktorý ovláda prah mapovania.

SVM používajúce sigmoidálne jadro bude ekvivalentné k dvojvrstvovej neurónovej sieti.

4. EXPERIMENTÁLNA ČASŤ

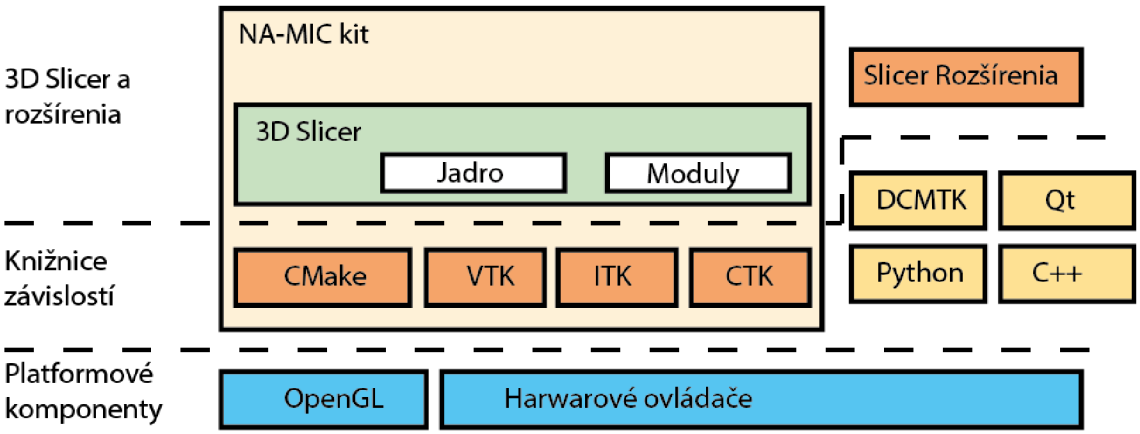
3D Slicer je vyvrcholením niekoľkých samostatných projektov, ktoré sa zaoberali vizualizáciou obrazu chirurgickej navigácie a grafického užívateľského rozhrania (GUI). David Gering predstavil prvý prototyp softwaru Slicer v jeho MIT práci, v roku 1999, na základe predchádzajúcich skúseností vo výskumných skupinách na MIT a SPL. Následne Steve Pieper prevzal rolu hlavného architekta a zahájil transformáciu 3D Slicer do profesionálnej podoby. Od roku 1999 Slicer bol pod neustálym vývojom, v SPL. Dnes je vyvíjaný najmä inžiniermi v úzkej spolupráci s vývojármi firiem Kitware, GE Global Research a za významného prispievania rastúcej Slicer komunity. Pôvodne bol software koncipovaný pre neurochirurgické poradenstvo, vizualizáciu a analýzu. V poslednom desaťročí sa Slicer vyvinul do integrovanej platformy, ktorá bola použitá v rôznych klinických a preklinických aplikáciách, rovnako aj na analýzu nelekárskych obrazov. [5]

Užívateľské rozhranie 3DSlicer aplikácie je zobrazené na obrázku 4.1. Okno pozostáva z viacerých sekcií. Ľavá časť obsahuje panel s nastaveniami pre aktuálne spustený modul aby užívateľ mohol s modulom komunikovať a upravovať parametre. Na pravej strane je niekoľko okien, ktoré zobrazujú pohľad z rôznych uhlov a celkový náhľad na 3D objekt.



Obrázok 4.1: 3D Slicer GUI

Architektúra 3D Slicer-u je modulárna a odstupňovaná (viď obr. 4.2). Na nižšej úrovni architektúry sú základné knižnice poskytované operačným systémom, ktoré nie sú balené s programom Slicer, ako napríklad OpenGL a hardwarové ovládače, ktoré umožňujú efektívne využitie grafických zdrojov hostiteľského systému. Na vyššej úrovni existujú jazyky predovšetkým C++ a Python, ale taktiež JavaScript a knižnice, ktoré poskytujú vyššiu úroveň funkčnosti a abstrakcie. Niektoré z ďalších knižníc používaných aplikáciou sú Qtframework a DICOM Toolkit (realizuje časti štandardu DICOM a interakciu s DICOM dátami a službami). Všetky externé závislosti 3D Slicer sú multiplatformne prenosné, distribuované pod licenciou plne kompatibilnou so Slicer, ktorá neobmedzuje ich používanie u komerčných alebo otvorených produktov. Vlastné funkcie je možné definovať za pomoci vykonávania externých modulov (Slicer Extensions). [6]



Obrázok 4.2: Architektúra 3D Slicer

3D Slicer samotný pozostáva z hlavného aplikačného jadra, Slicer modulov a Slicer rozšírení. Jadro implementuje užívateľské rozhranie, podporu pre vstupnú/výstupnú vizualizáciu a možnosť napojiť a rozšíriť program o ďalšie moduly. Veľké možnosti spolupráce s externými doplnkami viedli k vybratiu tohto softwaru, ako základného systému pre túto prácu. [6]

Individuálne rozšírenia môžu byť nezávislé, alebo môžu spolupracovať s ďalšími modulmi, ako napríklad s modulom, ktorý vykonáva segmentačnú funkcionalitu. Tento modul môže závisieť na module, ktorý renderuje obraz, aby umožnil 3D vizualizáciu segmentovanej štruktúry. Slicer moduly sú balíčkované spolu so Slicer distribúciou. Slicer rozšírenia sú externé doplnky, ktoré môže užívateľ ručne pridať a spustiť, podobne ako rozšírenia do webových prehliadačov. Tento spôsob externej spolupráce umožňuje zdieľanie aj napriek obmedzeniam licencií, prípadne ďalšími obmedzeniami. Rozšírenia môžu obsahovať viacero modulov.

Základné zabudované moduly môžeme rozdeliť do niekoľkých kategórií:

- Filtračné moduly: nástroje používané pri predspracovanie obrazu a obsahujú rôzne filtračné mechanizmy.

- Registračné moduly: nástroje, ktoré umožňujú priestorové usporiadanie snímok na základe rozličných parametrov.

- Segmentačné moduly: nástroje, ktoré oddeľujú jednotlivé subregióny v dátach, na základe určitých funkcií. Rovnako podporujú interaktívne, ale aj automatické metódy.

- Povrchové moduly: nástroje potrebné pri vytváraní a manipulácii s triangulovanými povrchovými modelmi.

Zbierka modulov sa pravidelne rozširuje o nové funkcionality, hlavne za významného prispievania komunity podporujúcej tento software.[5]

Moduly, ktoré je možné pridať do programu 3D Slicer môžeme rozdeliť do troch typov, CLI modul, zabudovaný modul a skriptovaný modul. Užívateľ nevidí rozdiel medzi modulmi, pretože v rozhraní sú rovnaké. Pre vývojára je typ modulu rozhodujúci v závislosti na type vstupu a parametroch pre daný modul.

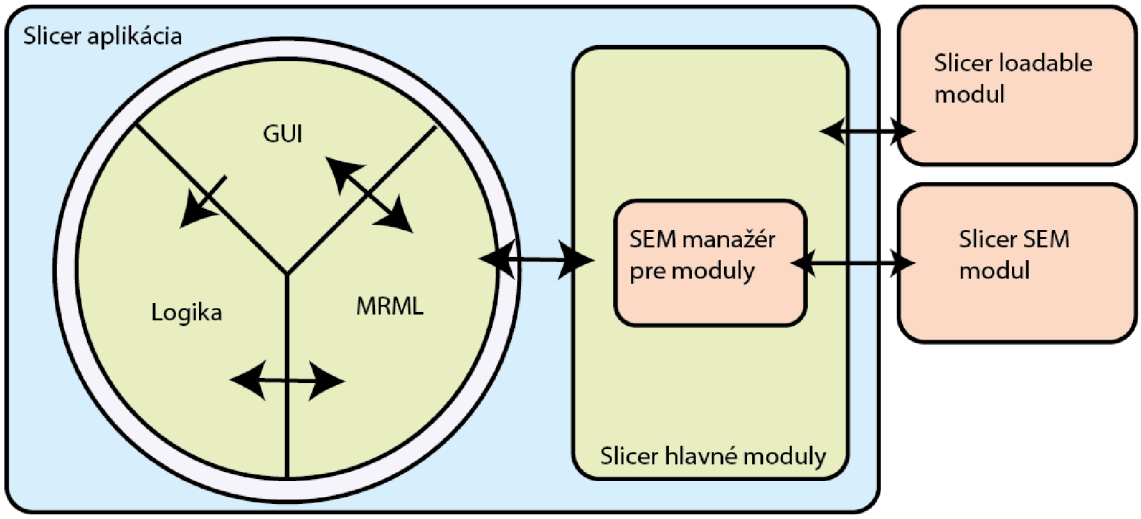
CLI modul je externý modul spustiteľný samostatne. Je limitovaný maximálnym počtom vstupných a výstupných parametrov. Vstupné argumenty môžu byť len jednoduché, pri ktorých modul neumožňuje užívateľskú interakciu. Jeho typickým znakom je jeho implementácia pomocou ITK. Užívateľské rozhranie je k modulu automaticky generované po pridaní do zoznamu 3D Slicer-u.

Skriptovaný modul je programovaný v jazyku Python. 3D Slicer obsahuje Python konzolu, v ktorej je možné vytvárať rýchle prototypy modulov. Keďže je Slicer z časti napísaný v Pythone, tento modul získava prístup k rôznym API, ktoré môže využívať, a preto je veľmi podobný vstavanému modulu.

Obidva typy modulov patria pod skupinu Slicer Execution Model (SEM) modulov. SEM implementuje jednoduchý prístup, ktorý nevyžaduje úplnú nutnosť poznať architektúru 3D Sliceru a nemá závislosť na zdrojovom kóde Sliceru.

Tretím typom je Slicer vstavaný modul. Tento modul je v jazyku C++ a priamo spolupracuje s 3D Slicerom. Načítava sa ako C++ zdieľaná knižnica pri spustení programu. Získava plnú kontrolu nad užívateľským rozhraním a všetkými Slicer internými aplikáciami. Je výhodný práve pre náročné výpočty a prácu s veľkým množstvom dát. Nevýhodou tohto modulu je, že vyžaduje kompiláciu spolu

so softwarom Slicer-u vzhľadom k použitej platforme pre ktorú software plánujeme používať. [6]



Obrázok 4.3: 3D Slicer exekučný model

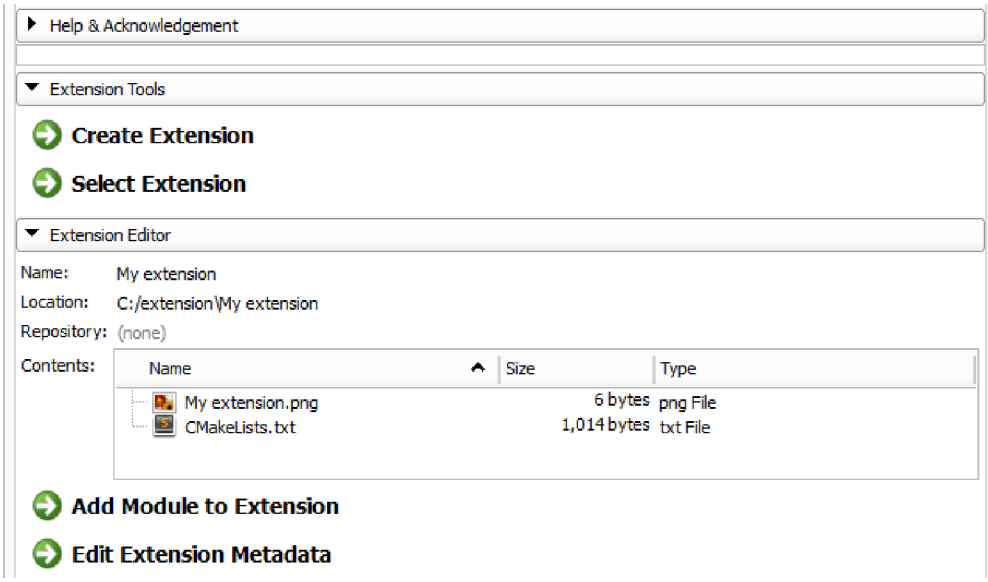
Táto práca je zameraná na možnosti segmentácie obrazu za pomoci softwaru 3D Slicer. Zabudované segmentačné moduly v programe sú rôzne, ako napríklad *EMSegmentEasy*, ktorý pracuje na základe algoritmu maximalizácie očakávania (EM). Je to iteratívny algoritmus opakujúci dva kroky. Prvý krok odhadne hodnoty nepozorovaných dát, a druhý maximalizuje dôveryhodnosť dát vzhľadom k dátam cez uvažované modely. Táto metóda vyžíva typ učenia bez učiteľa. Modul *Simple Region Growing Segmentation* využíva metódy narastania oblastí. Metóda má výhodu hlavne v obrazoch, kde je výrazný výskyt šumu a kde sa hranice určujú veľmi obtiažne. V programe sa však nevyskytuje modul, ani funkcia, ktorá by umožňovala segmentáciu na základe vstupu viacerých obrazov, ako napríklad T_1 a T_2 zároveň. Touto segmentáciou by bolo možné zvýšiť presnosť cielenej segmentácie a znížiť tak chybovosť.

4.1 TVORBA ZÁKLADU SKRIPTOVANÉHO ROZŠÍRENIA

Pre vytváranie rozšírení je potrebné mať vopred pripravené vývojárske prostredie, ktoré musí spĺňať určité podmienky. Posledná verzia programu Slicer obsahuje užívateľsky prívetivé nástroje pre vývojárov, ktorí chcú vytvárať rozšírenia. Zapnutie tohto módu je možné v nastaveniach, sekcia *Developer*, voľba *Enable developer mode* a *Enable QtTesting*. Po zapnutí developerského módu sa po reštarte aplikácie sprístupnia viaceré nástroje. Jedným z nich je *Extension Wizard*, ktorý umožňuje vytváranie základnej štruktúry budúceho rozšírenia.

Veľmi užitočný je aj nástroj *Python Interactor*, dostupný z hlavného okna Slicer aplikácie, ktorý umožní zobrazenie chybových hlásení a ponúka jednoduchú python konzolu, pomocou ktorej je možné otestovať časti kódu, prípadne funkčnosť modulu

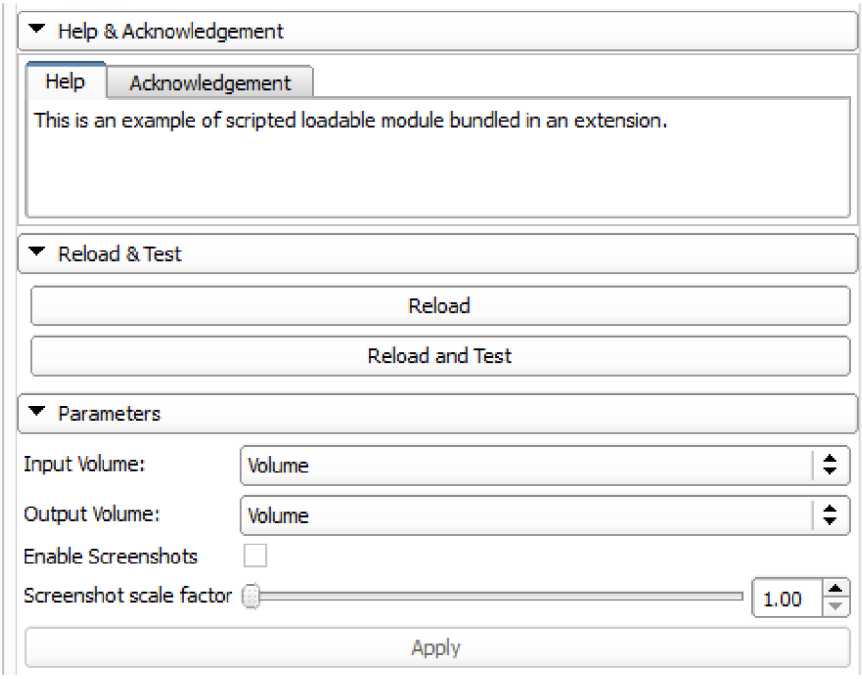
Vytvorenie rozšírenia pomocou *Extension Wizard* je intuitívne (viď obrázok 4.4). Prvým krokom je vytvorenie prototypu rozšírenia. Na výber je typ *default* alebo *superbuild*. Ďalšou voľbou je umiestenie nášho rozšírenia. Vhodný je prázdny priečinok bez súborov, ktoré by mohol program Slicer načítať naviac. Po potvrdení modul vygeneruje základnú adresárovú štruktúru a zobrazí panel pre vyplnenie metadát a popisu novo vytvoreného rozšírenia. Tieto údaje slúžia k priradeniu rozšírenia do ponuky Slicer-u, základný popis, menovanie vývojárov prípadne iné. Metadáta je možné kedykoľvek zmeniť pomocou voľby *Edit Extension Metadata*.



Obrázok 4.4: Rozhranie Extension Wizard modulu

Prázdne rozšírenie však samo nevykonáva žiadnu funkciu, zapuzdruje rôzne moduly do jedného celku s ktorým je možné pracovať a distribuovať na iné zariadenia. Voľba *Add Module to Extension* ponúka tri možnosti. Je v nej zahrnuté aký modul je možné pridať do rozšírenia, skriptovaný, CLI alebo vstavaný. V závislosti na výbere sa následne vygenerujú základné zdrojové kódy pre fungovanie modulu. Tieto kódy je možné ďalej upravovať a vytvoriť tak plnohodnotne rozšírenie programu 3D Slicer.

Po pridaní rozšírenia do ciest ktoré kontroluje program 3D Slicer je možné rozšírenie začať používať. Základné užívateľské rozhranie obsahuje vygenerované voľby pre výber vstupných, výstupných dát a tlačidlo na vykonanie úlohy (viď obrázok 4.5).



Obrázok 4.5: Rozhranie novovytvoreného modulu

Pre túto prácu sme zvolili skriptovaný modul v jazyku Python. Zdrojový kód modulu je v jednom súbore s príponou *.py*, kde sú definované triedy a metódy, ktoré sa volajú z užívateľského rozhrania.

Základná štruktúra skriptovaného modulu je hlavná zložka - rozšírenie s podzložkami - modulmi a súbormi vyjadrujúcimi vzťah medzi nimi:

\Extension\CMakeLists.txt	hlavný CMake súbor
\Extension\SVM	adresár dát skriptovaného modulu
\Extension\SVM.png	ikona rozšírenia
\Extension\SVM\CMakeLists.txt	CMake súbor pre skriptovaný modul
\Extension\SVM\Resources\Icons\SVM.png	ikona modulu
\Extension\SVM\SVM.py	zdrojový kód modulu

Základná štruktúra kódu skriptovaného modulu sa delí na inicializačnú časť, widget časť a logickú časť. Inicializačná časť je definovaná triedou *ScriptedLoadableModule*. Využíva základné prvky zo štruktúry programu 3D Slicer. Hlavnou časťou je definícia metadát, kde sú určené všetky základné popisy a umiestenie v užívateľskom rozhraní programu. V tejto triede sú automaticky definované časti z grafického rozhrania, ktoré obsahujú nápovedu a funkciu znovu načítania a testovania modulu. Časti sú generované systémom a nie je možné ich funkčnosť výrazne ovplyvňovať.

Časť Widget je definovaná triedou *ScriptedLoadableModuleWidget*. V nej sa definujú metódy pre užívateľské rozhranie modulu. Patrí sem definícia pre tlačidlá, interakcie medzi nimi a prepojenia s rozhraniami ďalších modulov. Obsah je plne pod správou vývojára modulu.

Logická časť je definovaná triedou *ScriptedLoadableModuleLogic*. Táto trieda implementuje všetky výpočtové operácie vykonávané daným modulom. Rozhranie by malo byť definované tak, aby bolo použiteľné aj externe. Trieda by mala byť použiteľná v ďalších moduloch a funkčná aj bez potreby vytvárania inštancie *widget* časti modulu. Trieda môže byť použitá aj ako testovacia, pričom metódy obsiahnuté v nej, môžu byť definované pre testovanie funkčnosti daného modulu.

Prídavnou časťou je trieda *ScriptedLoadableModuleTest*, ktorá je určená pre testovací prípad použitia skriptovaného modulu.

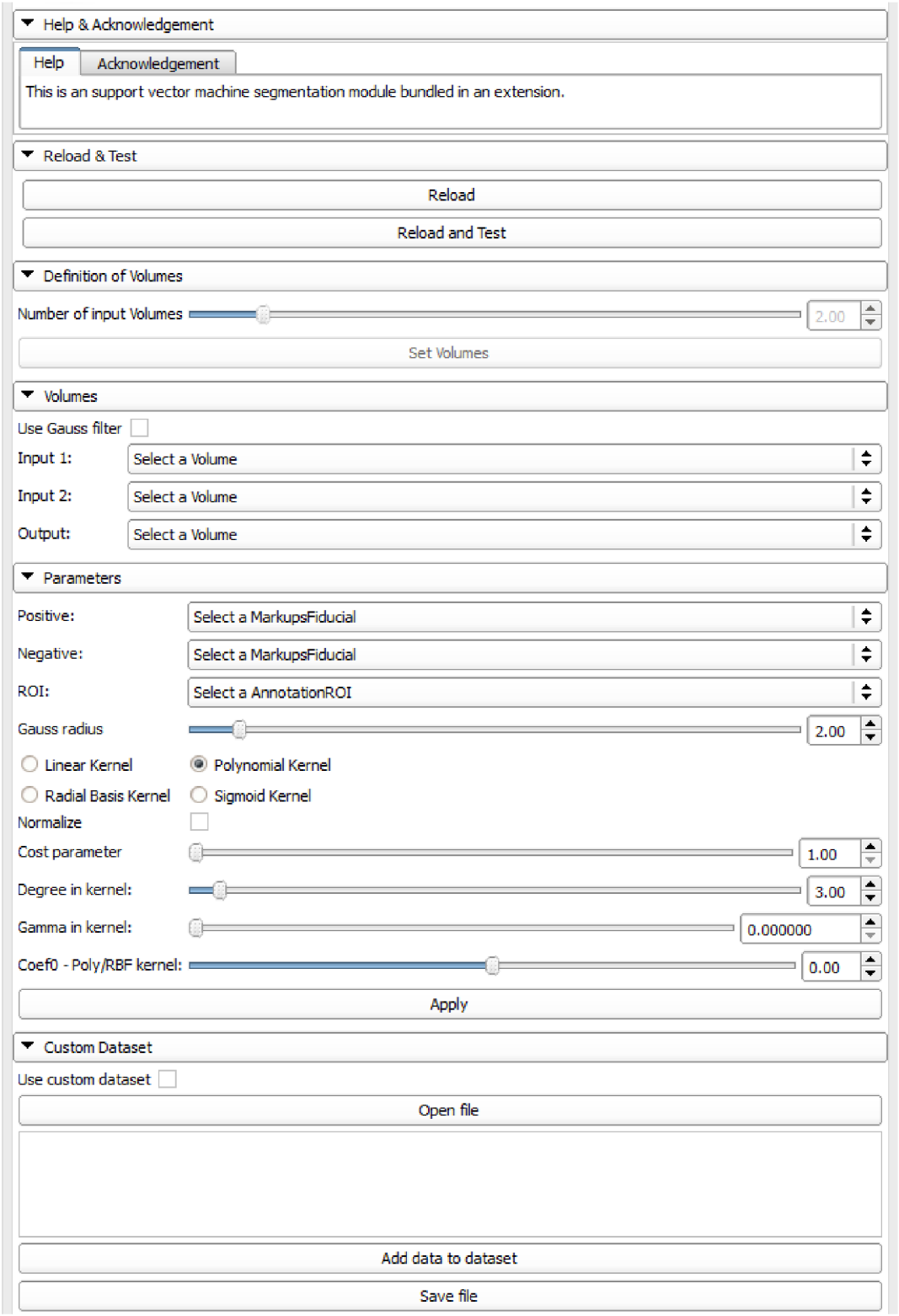
4.2 POPIS ROZHRANIA GUI A PARAMETROV ROZŠÍRENIA

V tejto časti je popísané rozšírenie programu 3D Slicer, ktoré umožňuje segmentovať MR obraz pomocou SVM metódy. Rozšírenie využíva skriptovaný modul napísaný v jazyku Python a knižnicu *libsvm*, ktorá implementuje vlastnú metódu SVM. Pre toto rozšírenie je potrebná verzia 3D Slicer 4.4.0 64 bitová, Python verzia 2.7.3 64 bitová a samotná knižnica *libsvm* taktiež vo verzii 64 bit pre Python.

Pre ideálny chod programu 3D Slicer je potrebné mať k dispozícii počítač s dostatočnou kapacitou pamäte RAM, optimálne 8 GB a viac, z dôvodu pamäťovej náročnosti spracovania 3D obrazových dát. Pri nedostatočnej veľkosti pamäte môže nastať situácia, kedy program využije všetky dostupné prostriedky a začne využívať odkladací priestor na disku, čo dramaticky spomalí chod segmentácie a celého počítača. Ďalej je nutné mať 64 bitový operačný systém. Na verzii systému nezáleží, pretože program 3D Slicer je multiplatformový a je schopný pracovať na systémoch Windows, Linux, OSX. Editácia rozšírenia je možná v akomkoľvek textovom editore, prípadne je možné použiť editor zameraný na jazyk Python. Ten umožňuje lepšie využívanie jeho možností, ako našepkávanie alebo automatickú opravu kódu. Odhaľovanie chýb v rozšírení je jednoducho riešiteľné len po spustení v programe 3D Slicer, pričom je možné využiť funkciu *Python Interactor* na zobrazenie chybových hlásení a výpis z konzoly. Grafické užívateľské rozhranie modulu je pevne spojené s programom 3D Slicer. Využíva univerzálnu knižnicu *Qt gui*, ktorá zabezpečuje chod užívateľského rozhrania cez všetky platformy.

Po načítaní programu 3D Slicer a pripojení modulu, je ho možné nájsť v štandardnej ponuke modulov. Užívateľské rozhranie je rozdelené na niekoľko častí.

Každá časť je graficky rozdelená pomocou skrývacích tlačidiel. Obrázok č. 4.6 ukazuje ponuku vytvoreného rozšírenia.



Obrázok 4.6: Rozhranie vytvoreného SVM modulu

Prvá časť ponuky obsahuje pomocníka, popis programu a základné informácie o licenciách k rozšíreniu.

Druhá časť obsahuje tlačidlá *Reload* a *Reload* and Test, pričom *Reload* zmaže všetky aktuálne nastavenia a premenné použité v rozšírení a nastaví ich do výchozích hodnôt. *Reload and test* tlačidlo má rovnakú funkciu ako tlačidlo *reload*, naviac však vykoná základný test funkčnosti rozšírenia.

Tretia časť už je zameraná na definíciu základných vstupov do programu. Z dôvodu používania segmentácie SVM, ktorá je viac parametrická, je potrebné definovať viacero vstupov - vstupných obrazov, z ktorých bude možné čerpať dáta do samotného spracovania. Rozšírenie je navrhnuté tak, aby bolo čo najviac univerzálne, tiež podporuje nastavenie až 10 vstupných obrazov. Výber počtu vstupných obrazov je potrebné potvrdiť tlačidlom *Set Volumes*. Je to z dôvodu, že program sa prispôsobí počtu vstupných obrazov a definuje dostatočné množstvo volieb pre ďalšie nastavenia.

Ďalšia časť sa zobrazí na základe nastavenia časti o počte vstupných obrazov. Obsahuje voľbu pre výber vstupných obrazov, výstupný obraz a možnosť pre doplnenie ku každému vybranému vstupnému obrazu jeho klon upravený rozmazaním pomocou funkcie gausián. Vstupné obrazy je možné vyberať len z dát, ktoré boli vopred načítané do programu 3D slicer. Vstupné dáta je možné importovať zo súborov DICOM prípadne Nifti. Všetky dáta musia byť upravené na rovnaký počet rezov, musia mať rovnaké rozlíšenie a musia byť registrované tak, aby používali rovnaký súradnicový systém. Je to nutné z dôvodu, aby program mal ku každému voxelu, ktorý bude spracovávať, adekvátne voxely z iných zdrojov obsahujúce ďalšie vstupné parametre. Registráciu môžeme vykonať napríklad pomocou modulu *Expert Automated Registration*, ktorý už program 3D Slicer obsahuje. Vytvorenie rovnakého počtu rezov a zabezpečenie správneho rozlíšenia obrazu je možné vykonať pomocou modulu *Resample scalar/vector/dwi volume*.

Časť s názvom Parameters obsahuje rôzne voľby pre nastavenie segmentácie pomocou SVM, výber jadier SVM a ich parametrov. Hlavnou časťou je výber pozitívnych a negatívnych bodov na základe ktorých sa bude trénovať SVM algoritmus. Pre urýchlenie finálnej klasifikácie je tu možnosť výberu ROI, ktoré umožní zmenšiť veľkosť prehľadávaného obrazu na nami určený rozsah. Pozitívne a negatívne body sú dátové sety voxelov ukazujúce na súradnice v 3D obraze, ktoré boli takto určené. Voľbu týchto bodov môžeme vykonať pomocou modulu *Markups*, a vytvoriť tak rôzne veľký set pozitívnych a negatívnych hodnôt. ROI je rovnako možné vytvoriť pomocou modulu *Markups*. V programe 3D Slicer je voľba ROI štandardne obmedzená pre zjednodušenie spracovania na veľkostne nastaviteľný kváder.

Ďalšou možnosťou je nastavenie veľkosti gaussového jadra, pomocou ktorého sa vytvárajú rozmazané klony obrazov. Rozsah veľkosti jadra je definovaný ako polomer jadra s hodnotami medzi 1 až 15, pričom preddefinovaná hodnota polomeru je 2, čo reprezentuje jadro o veľkosti 5 × 5 × 5 voxelov.

V ponuke je ďalej na výber z 4 možností SVM jadra, pomocou ktorého bude vykonávaná klasifikácia dát, a to lineárne jadro, polynomiálne jadro, RBF jadro a sigmoidálne jadro. Tento výber je veľmi dôležitý. Na základe výberu jadra sa použijú rozdielne funkcie a výrazne ovplyvňujú výsledok klasifikácie obrazových dát. Ďalšou možnosťou na výber je použitie normalizácie. Táto funkcia umožní transformáciu vstupných dát, ktoré môžu naberať rôzne vysoké hodnoty, na hodnoty porovnateľné s šedotonovým zobrazením obrazu.

Cost parameter je jedným z vlastných parametrov SVM. Používa sa ako optimalizácia SVM. Nastavuje sa tu penalizačný koeficient, ktorý zabraňuje nesprávnej klasifikácií pre každú vstupnú hodnotu v trénovacej sekvencii SVM viď vzorec 3.14. Pri nastavení veľkých hodnôt tohto koeficientu optimalizačný algoritmus vyberie menšiu vzdialenosť k rozdeľovacej rovine a SVM sa potom snaží mať všetky hodnoty správne klasifikované. Naopak, ak bude nastavená malá hodnota parametru, optimalizátor môže niektoré body vymykajúce sa štandardu ignorovať a priradiť im zlý klasifikátor. V prípade spracovania MR dát je vhodné počítať s tým, že niektoré body môžu dosahovať výrazne rozdielne hodnoty od štandardných z rovnakého setu, a voliť tak nižší koeficient. Štandardne je preddefinovaný *Cost parameter* ako 1.

Nastavenie *degree in kernel* sa používa u polynomiálneho SVM jadra a znamená na aký stupeň bude umocnená celá funkcia. Štandardná hodnota je 3, čo znamená, že bude použité polynomiálne jadro tretieho rádu. Vo vzorci 3.20 reprezentuje parameter d a môže mať len hodnoty z oboru prirodzených čísel. Ak $d = 1$ toto jadro sa bude správať ako lineárne jadro +1.

Parameter *gamma in kernel* sa používa hlavne pri RBF a sigmoidálnom jadre. U RBF reprezentuje tento parameter hodnotu γ vo vzorci 3.19, ktorý vyjadruje reguláciu rádiusu. Čím nižšia hodnota, tým je funkcia citlivejšia na zmeny. Naberá len kladné hodnoty. V sigmoidálnom jadre tento parameter γ (vzorec 3.21) upravuje hodnoty vstupných dát. Preddefinovaná hodnota pre tento parameter je 1/počet vlastností jedného prvku.

Parameter *Coef0* sa používa u sigmoidálneho jadra (vzorec 3.21), prípadne môže mierne upravovať polynomiálne jadro. Tento parameter ovláda prah mapovania. Preddefinovaná hodnota je 1.

Potvrdenie nastavení a vykonanie SVM klasifikácie pomocou nadefinovaných parametrov a vstupných dát je možné pomocou tlačidla *Apply*. Doba spracovania dát závisí od veľkosti vstupných dát, počtu vstupných vlastností, veľkosti ROI skúmanej oblasti, tiež závisí od počtu trénovacích dát a typu SVM, podľa ktorého sa rozlišuje doba trénovania modelu SVM.

Doplnková súčasť rozšírenia *Custom dataset* obsahuje možnosť vytvoriť databázu trénovacích dát skladajúcu sa z viacerých vstupných obrazov. Účelom tejto vlastnosti je vylepšenie klasifikačnej schopnosti tejto metódy použitím rozšíreného dátového setu trénovacích hodnôt. Pri použití vhodných trénovacích dát nie je potrebné na testovacom obraze vytvárať dátový set pozitívnych a negatívnych klasifikátorov.

Funkcia sa povolí zatrhnutím voľby *Use custom dataset*. Hlavnou častou je okno s náhľadom na trénovacie dáta. Tieto dáta je možné upravovať podobne ako v textovom editore. Voľba *Open file* umožní načítať ľubovoľný dátový set trénovacích hodnôt z textového súboru. Po načítaní súboru sa prepíše obsah textového okna dátami z vstupného súboru. Voľba *Save file* umožní aktuálnu databázu uložiť do súboru pre ďalšie použitie.

Použitie voľby *Add data to dataset* je využiteľné pri vytváraní databázy trénovacích dát pre metódu SVM. Voľba využije vstupné obrazy z časti *Volumes* a k nim vyžaduje trénovacie klasifikátory z vstupných polí *Positive* a *Negative*. Zo vstupných dát vyberie trénovacie dáta a pridá ich na koniec aktuálnej databázy.

Pre použitie trénovacej databázy na trénovanie modelu SVM a klasifikácie na aktuálnych testovacích dátach stačí použiť tlačidlo *Apply* a pritom mať zatrhnutú voľbu *Use custom dataset*. V tomto prípade sa nevytvára nový trénovací set pre tvorbu modelu, ale použije sa databáza klasifikovaných trénovacích dát. Parametre SVM a testovacie dáta sa použijú rovnaké ako pri štandardnom použití tohto rozšírenia.

4.3 IMPLEMENTÁCIA ROZŠÍRENIA DO PROGRAMU 3D SLICER

Obsahom tejto kapitoly je popis kódu programu a rozbor použitých funkcií a metód. Z dôvodu, že program 3D Slicer nie je jednoducho rozšíriteľný o rôzne doplnkové knižnice, bolo nutné použiť mierne odlišný prístup od štandardného použitia vytvárania skriptovaných rozšírení. Využitie knižnice *libsvm* vyžaduje pokročilé funkcie z knižnice ctypes, ktorá nie je úplne obsiahnutá v štandardnej zostave programu 3D Slicer. Z toho dôvodu je nutná inštalácia systémového Pythonu, ktorý tieto knižnice štandardne obsahuje v plnej verzii. Verzia Pythonu musí byť 64 bitová, pretože program 3D Slicer štandardne neexistuje vo verzii 32bitovej, a musí byť zachovaná spoločná kompatibilita pre kombinované použitie. Kombinovanie interného a systémového python prekladača v zdrojovom kóde modulu nám zabezpečí funkčnosť programu a vyhne sa

nekompatibilite s internými modulmi programu 3D Slicer. Nevýhodou tohto riešenia je nutnosť nastaviť v zdrojovom kóde modulu cestu k systémovej inštalácii jazyka Python:

```
sys.path.insert(0, "C:/Python27")
```

Knižnica *libsvm* (A Library for Support Vector Machines) je voľne dostupný integrovaný sofware pre klasifikáciu pomocou podporných vektorov. Tento systém je dostupný vrátane zdrojových kódov, binárnych súborov pre Windows, aj pomocných skriptov v jazyku Python.

Základne funkcie použité v implementácii do modulu sú:

```
SVMproblem = svm_problem(y, x)
```

Funkcia vytvorí dátovú štruktúru potrebnú pre spracovanie SVM, za parametrom *x* sú hodnoty vlastnosti daného prvku v štruktúre typu list. Parameter y klasifikuje vlastnosti prvku na základe ktorých bude určená rozdeľovacia nadrovina. Dátový typ musí byť integer alebo double.

```
SVMparameter = svm_parameter(parameters)
```

Vytvára štruktúru obsahujúcu parametre potrebné pre správne nastavenie klasifikačného algoritmu. Základné nastavenia algoritmu sú rozhodujúce pre korektné spracovanie požadovaných dát. Ovplyvňovať tieto parametre je možné z užívateľského rozhrania v plnom rozsahu. Pokiaľ nie je vyplnená žiadna hodnota, použijú sa východzie hodnoty.

```
model = svm_train(SVMproblem, SVMparameter)
```

Tato funkcia vytvára model SVM na základe vstupu z dátových štruktúr *SVMproblem* a *SVMparameter*. Výsledkom trénovania SVM modelu je rozhodujúca klasifikačná nadrovina, ktorá môže byť použitá na klasifikovanie testovacích dát.

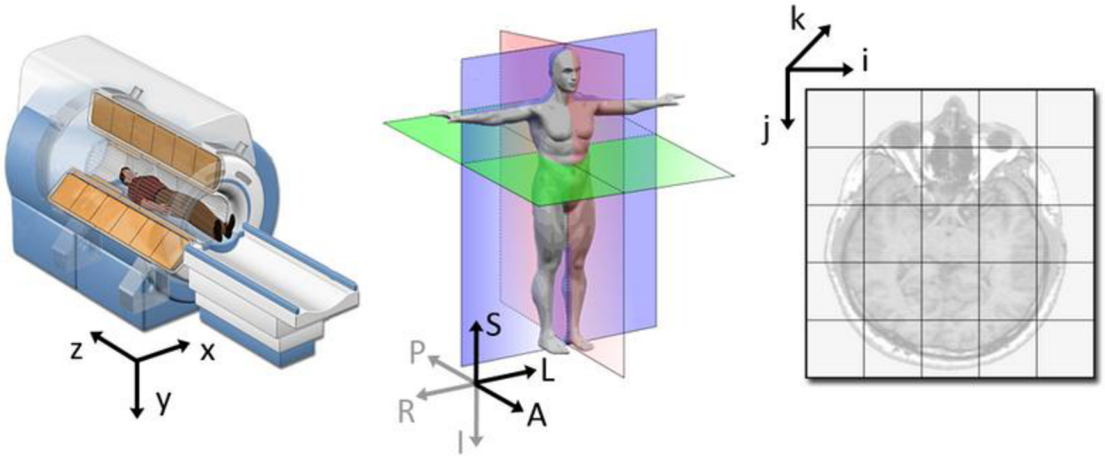
```
p_labels, p_acc, p_vals = svm_predict(y, x, m)
```

Funkcia využíva natrénovaný model *m* a vstupné dáta *x* a *y*. *X* sú atribúty jedného prvku v dátovej štruktúre typu list, *y* je voliteľný a použiteľný pri testovaní kvality klasifikácie SVM. Výstupom je set klasifikovaných dát.

Problematika koordinačných systémov

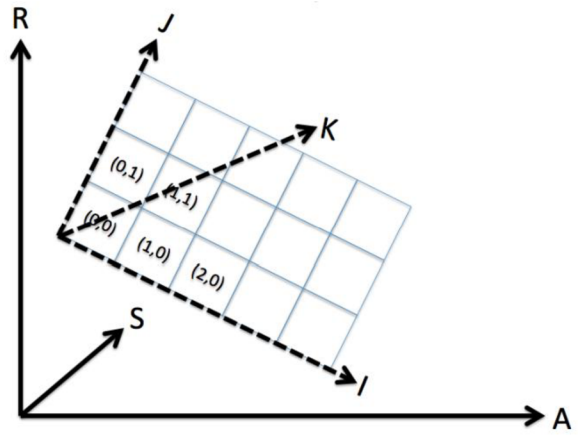
3D Slicer využíva viacero koordinačných systémov. Ako hlavný systém používa Right Anterior Superior systém (RAS). Je to najdôležitejší modelový koordinačný systém pre použitie v medicíne v anatomickom priestore. Tento priestor sa skladá z troch rovín, ktoré opisujú štandardnú anatomickú pozíciu človeka. Axiálna rovina je paralelná so

zemou a oddeľuje hlavu – *Superior*, od nôh - *Inferior*. Koronálna rovina je kolmá na zem a oddeľuje prednú časť - *Anterior*, od zadnej časti - *Posterior*. Sagitálna rovina oddeľuje ľavú stranu od pravej (viď obrázok 4.7). Tento koordinačný systém je kontinuálny trojdimenzionálny priestor, v ktorom je navzorkovaný obraz. Je bežné definovať priestor vzhľadom k ľudskému telu, ktorého je napríklad mozog naskenovaný. Mriežka 3D priestoru je definovaná okolo anatomických rovín. Pre prácu s VTK obrazmi používa IJK koordinačný systém. Tento obrazový koordinačný systém popisuje ako bol obraz skenovaný, vzhľadom k anatómii. Medicínske skenery vytvárajú pravidelné obdĺžnikové polia bodov a buniek, ktoré majú začiatok v ľavom hornom rohu. Neoddeliteľnou súčasťou je definovanie vzdialenosti medzi každým bodom. Je to hlavne z toho dôvodu, že každá os môže mať iné vzdialenosti medzi dvoma po sebe idúcimi bodmi. Tretí koordinačný systém využíva typický Kartézsky koordinačný model, v ktorom je objekt umiestnený. Každý model ma vlastný koordinačný systém, avšak je tu len jeden obecný systém pre definíciu pozície a orientácie modelu.



Obrázok 4.7: Koordinačné systémy používané v medicíne

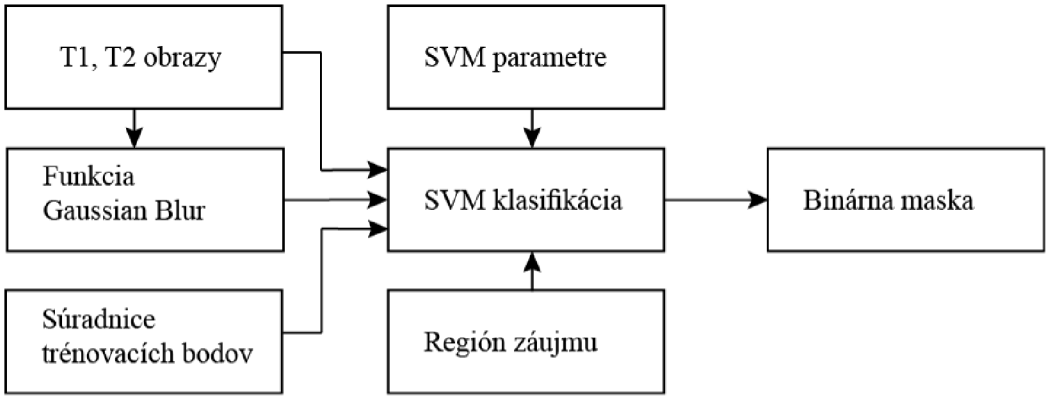
Aby bolo možné korektne spracovávať informácie obsiahnuté v obrazoch a aplikovať ich v programe Slicer, je potrebné ich previesť na rovnaké koordinácie (viď obrázok 4.8). Na toto prevedenie sa používa transformačná matica, ktorú môžeme vytvoriť pomocou funkcie *vtk.vtkMatrix4x4()*. Túto maticu potom môžeme použiť na konverziu medzi oboma koordinačnými systémami. Pred touto operáciou ju však treba naplniť hodnotami. Rozlišujeme smer, ktorým koordináty prevádzame, a preto pre vytvorenie matice na prevod IJK do RAS, používame funkciu *GetIJKToRASMatrix()* a naopak *GetRASToIJKMatrix*. Prevod RAS na IJK je komplikovanejší, a preto je dodatočne nutné vykonať operáciu prenásobenia *outIJK = ras2ijkmatrix.MultiplyPoint(inRAS)*.



Obrázok 4.8: Vzťah medzi IJK a RAS koordinátmi

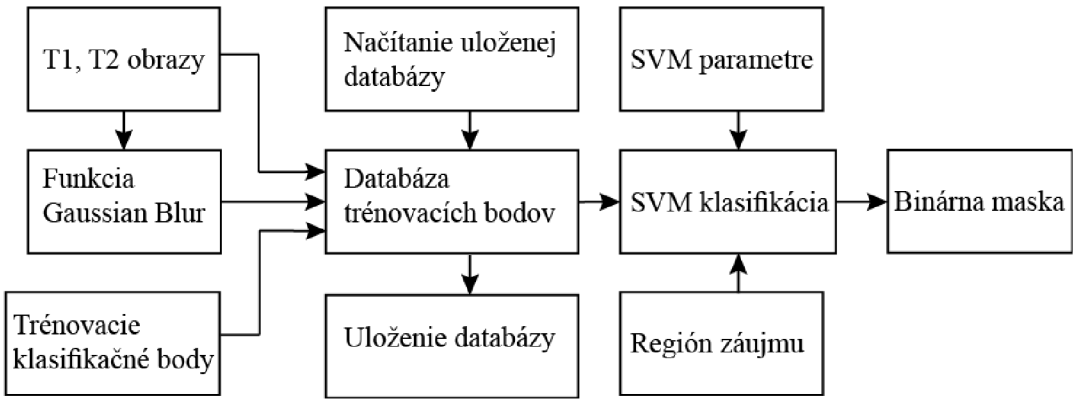
4.3.1 Metodika rozširovacieho modulu programu

Rozširovací modul je založený na binárnej klasifikácii multiparametrických dát metódou SVM. V prvom kroku je nutné mať obrazové dáta správne registrované. Podľa počtu vstupných obrazov sa upravuje funkčnosť rozšírenia. Tieto obrazové dáta predstavujú hlavnú vstupnú zložku pre ďalšie spracovanie. Na základe užívateľskej voľby sa k týmto obrazom po spustení doplnia ich deriváty upravené pomocou funkcie gaussian blur. Základná schéma funkčnosti rozšírenia je zobrazená na obrázku 4.9. Neoddeliteľnou súčasťou vstupných dát je označenie klasifikačných bodov a regiónu záujmu. Klasifikačné body slúžia ako trénovací set metódy SVM. Región záujmu obmedzuje finálnu klasifikáciu na túto oblasť a zrýchľuje celú operáciu. Celé trénovanie modelu SVM môže byť ovplyvnené výberom parametrov klasifikačných jadier. Výstupom z klasifikácie je binárny obraz, ktorého obsahom sú klasifikované vstupné dáta. Prístup metódy SVM hľadá maximálnu vzdialenosť medzi hyperrovinou a dátami oboch testovacích skupín dát. Dobré segmentačné výsledky môžu byť dosiahnuté ak parametre SVM modelu sú vhodne zvolené na základe kvality vstupných obrazov.



Obrázok 4.9: Základná schéma modulu SVM

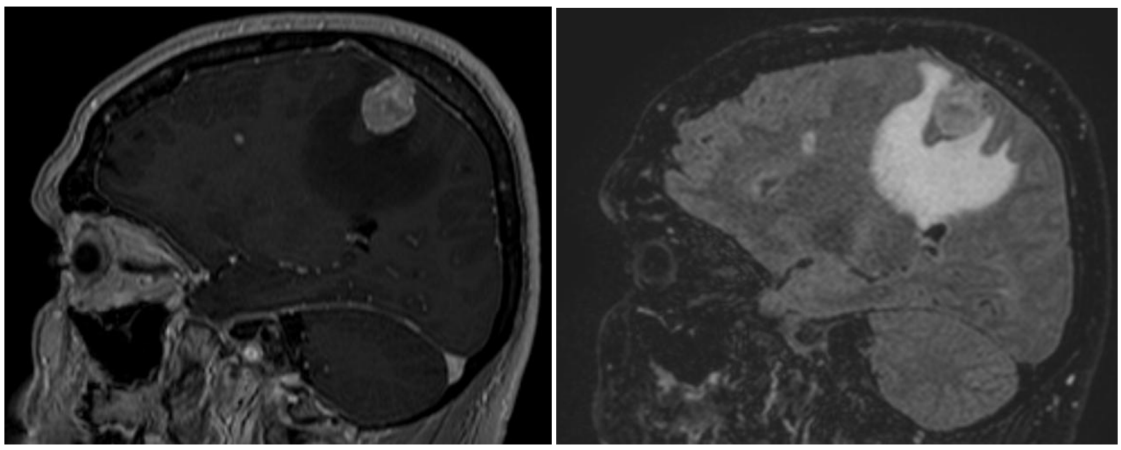
SVM rozšírenie programu 3D Slicer obsahuje voliteľnú variantu trénovania modelu. Trénovací dátový set je možné vytvoriť z viacerých vstupných obrazových setov iných pacientov. Rozšírenie trénovacieho setu o správne doplnkové dáta vedie k zlepšeniu klasifikačnej schopnosti metódy SVM. Databázu trénovacích dát je možné vytvárať a viacnásobne upravovať.



Obrázok 4.10: Rozšírenie modulu o databázu trénovacích dát

Príklad použitia vytvoreného modulu

Je potrebné mať k dispozícií MR obrazy typu T_1 a T_2. Obrazy musia byť upravené na rovnaké rozlíšenie voxelov. Ďalej je potrebné mať pripravené dva sety označených dát z modulu programu Slicer - Markups. Tieto sety reprezentujú trénovaciu množinu metódy SVM. Je potrebné mať pripravenú množinu pozitívnych aj negatívnych dát. To znamená mať označené body ktoré sú podľa nás správne v regióne záujmu a ktoré nie.

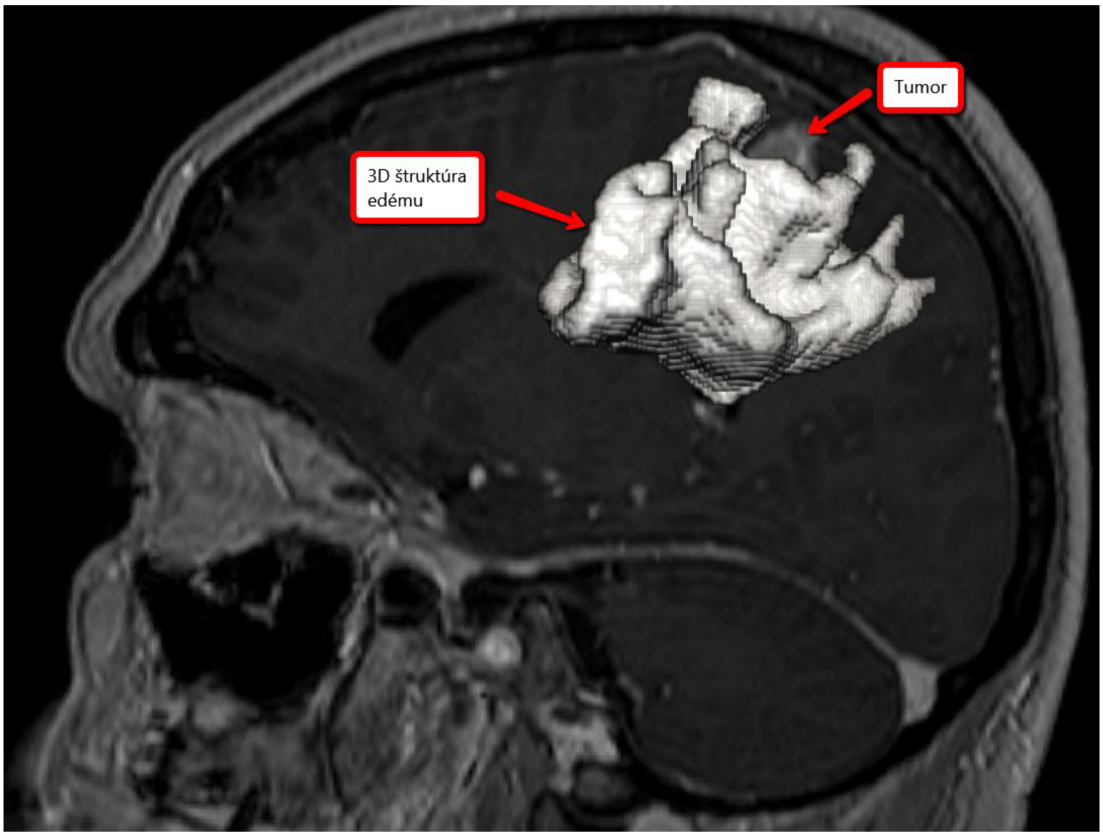


Obrázok 4.11: Vstupné obrazy do SVM modulu

Po otvorení SVM modulu v ponuke pre výber počtu vstupných dát ponecháme 2 a potvrdíme voľbou *Set Volumes*. V okne zvolíme vstupné dáta naše vstupné obrazy. Pre výstupný obraz vyberieme vo voľbe *Output* vytvoriť novú položku. Pre dosiahnutie lepších výsledkov zatrhneme voľbu *Use Gauss filter*.

V sekcii *Parameters* zvolíme pozitívne a negatívne označené sety ktoré sme dopredu pripravili. V *ROI* zvolíme vytvoriť nové ROI alebo vyberieme z dostupných ktoré už boli vytvorené cez modul Markups - ROI. *Gauss radius* voľbu ponecháme na štandardnej hodnote 2.

Zvolíme SVM jadro *Radial Basis Kernel*. *Cost* paramater ponecháme na hodnote 1. Upravujeme v prípade že segmentácia ma veľa nesprávne klasifikovaných bodov. Dôležitá je voľba *Gamma in kernel*, nastavíme na vyššiu citlivosť, napríklad hodnotu 0,000001. Ostatné nastavenia ponecháme na štandardných nastaveniach. Po kliknutí na *Apply* sa spustí segmentácia a následne zobrazí výsledný segmentovaný obraz ako maska viď obrázok 4.12.

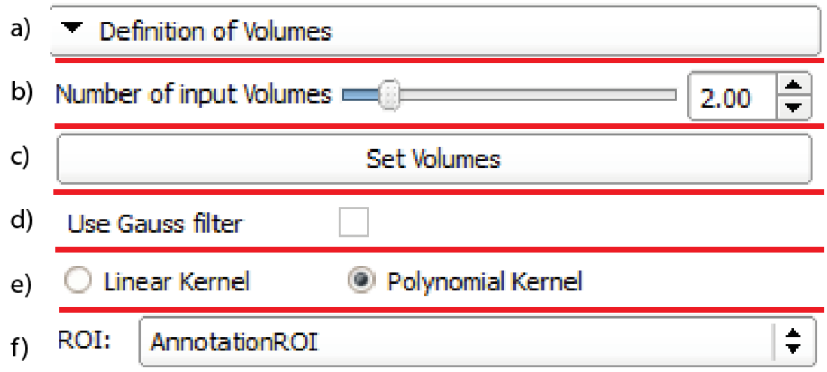


Obrázok 4.12: Výsledná segmentovaná maska edému prevedená do 3D priestoru

V ukážke sme použili trénovacie dáta ktoré určovali príslušnosť k tkanivu typu edém. Preto v segmentovanom obraze je vidieť štruktúru edému. Na obrázku je taktiež vidieť ako edém obklopuje tumor.

4.3.2 Implementácia grafického rozhrania

Obsahom tejto kapitoly je popis implementácie grafických prvkov v jazyku Python s využitím Qt frameworku, a ich komunikácia s logickou časťou rozšírenia. Pre prehľadnosť v rozšírení a oddelenie ovládacích prvkov, je vhodné ich rozdeliť do skupín. Na tento problém môže byť použité rolovacie tlačidlo, ktoré zapuzdruje ďalšie ovládacie prvky a urobí tak GUI dynamickým (obrázok 4.13a).



Obrázok 4.13: Ukážka definovaných grafických prvkov

```
basicCollapsibleButton = ctk.ctkCollapsibleButton()
basicCollapsibleButton.text = "Definition of Volumes"
self.layout.addWidget(basicCollapsibleButton)
basicFormLayout = qt.QFormLayout(basicCollapsibleButton)
```

Táto časť kódu popisuje vytvorenie skrývacieho tlačidla, definuje prvok *basicCollapsibleButton* s textom "Definition of Volumes". Tento prvok sa pridá do hlavného usporiadania *self.layout*, a pretože môže zapuzdrovať ďalšie prvky, vytvorí sa pre toto tlačidlo ďalšie usporiadanie vo vnútri skrývacieho tlačidla s názvom *basicFormLayout*.

Do tohto vnútorného rozdelenia môžeme vkladať ďalšie prvky, napríklad posuvný nastavovač hodnôt. Využíva sa tu knižnica CTK. (obrázok 4.13b)

```
self.numberofvolumes = ctk.ctkSliderWidget()
self.numberofvolumes.singleStep = 1.0
self.numberofvolumes.minimum = 1.0
self.numberofvolumes.maximum = 10.0
self.numberofvolumes.value = 2.0
self.numberofvolumes.setToolTip("Number of input Volumes")
basicFormLayout.addRow("Number of input Volumes", self.numberofvolumes)
```

Nastavenia tohto prvku sú minimum, maximum, štandardná hodnota a minimálny krok. Voliteľne je možné doplniť počet desatinných miest. Celý prvok je potrebne pridať do usporiadania. Voľbou addRow pridáme riadok do usporiadania a vedľa seba môžeme umiestniť dva prvky, v tomto prípade je to text s názvom a posuvný nastavovač.

Potvrdenie voľby je potrebne vykonať základným tlačidlom. Využíva štandardné prvky z Qt frameworku (obrázok 4.13c).

```
self.setvolumes = qt.QPushButton("Set Volumes")
self.setvolumes.toolTip = "Set Volumes."
self.setvolumes.enabled = True
basicFormLayout.addRow(self.setvolumes)
```

Tlačidlu je možné, okrem názvu, definovať pomocníka, ktorý sa zobrazí pri ukázaní kurzorom na tlačidlo. Parametre ako podmienené povolenie, alebo zakázanie funkčnosti tlačidla sú taktiež možné. Na základe kliknutia na tlačidlo je možné nadefinovať beh ďalších funkcií. Spúšťač je potrebné definovať pre konkrétnu akciu.

```
self.setvolumes.connect('clicked(bool)', self.onsetvolumes)
```

Tento spúšťač reaguje na kliknutie na tlačidlo *setvolumes*, a zavolá funkciu *onsetvolumes*, ktorá obsahuje ďalší kód programu.

Ďalším z jednoduchých ovládacích prvkov je checkbox, ktorý využíva QtFramework (obrázok 4.13d). Užívateľova voľba ovplyvňuje ďalší beh programu. V tomto prípade je štandardne voľba vypnutá.

```
self.dogauss = qt.QCheckBox()
self.dogauss.checked = 0
self.dogauss.setToolTip("Use gauss filter to get better classification")
volumesFormLayout.addRow("Use Gauss filter", self.dogauss)
```

Tradičným parametrom je nastavenie pomocníka a umiestnenie v rozložení modulu.

Neoddeliteľnou súčasťou grafického rozhrania je prepínač medzi viacerými voľbami (obrázok 4.13e). Z týchto volieb je možné vybrať len jednu. V nasledujúcom kóde vyberáme medzi nastavením jadra. Predvolená voľba je polynomiálne jadro. Z dôvodu využitia QtFramework sú tu dostupné rovnaké nastavenia ako u predchádzajúcich.

```
self.radio = qt.QRadioButton("Linear Kernel")
self.radio.toolTip = "Set kernel."
```

```
self.radio.enabled = True
self.radio2 = qt.QRadioButton("Polynomial Kernel")
self.radio2.toolTip = "Set kernel."
self.radio2.checked = True
self.radio2.enabled = True
parametersFormLayout.addRow(self.radio, self.radio2)
```

3D Slicer využíva tzv. MRML koncept. Je to systém dátových kontajnerov, ktoré zapuzdrujú rôzne štruktúry. Každý kontajner má unikátny identifikátor. Pre zobrazenie v aplikácii sa používa MRML Scene kontajner, ktorý je len jeden pre aplikáciu. Dátové kontajnery môžu obsahovať obrazové dáta zo systému DICOM, označenie vlastností rôznych bodov, či výber regiónu záujmu. V grafickom prostredí sa často využíva výber z dostupných MRML kontajnerov (obrázok 4.13f). Tento ovládací prvok je súčasťou knižnice Slicer.

```
self.inputSelectorLabel = qt.QLabel()
self.inputSelectorLabel.setText("Input 1: ")
self.inputSelector = slicer.qMRMLNodeComboBox()
```

Pretože je tento prvok univerzálny, je potrebné filtrovať medzi rôznymi typmi MRML kontajnerov. Na to slúži obmedzenie výberu pomocou *nodeTypes*. Výberom *vtkMRMLScalarVolumeNode* zvolíme len obrazové štruktúry. Výberom *vtkMRMLMarkupsFiducialNode* volíme značkovaciu štruktúru a voľbou *vtkMRMLAnnotationROINode* definujeme výber z anotačnej štruktúry.

```
self.inputSelector.nodeTypes = ( ("vtkMRMLScalarVolumeNode"), "" )
self.inputSelector.nodeTypes = ( "vtkMRMLMarkupsFiducialNode", "" )
self.inputSelector.nodeTypes = ['vtkMRMLAnnotationROINode']
```

Tento prvok umožňuje okrem výberu aj vytvorenie prázdnych štruktúr, ich premenovanie či zmazanie. Tieto voľby môžeme obmedziť. Je potrebné definovať MRML scénu aby bolo jasné, s ktorými dátami má pracovať. Ako aj u predchádzajúcich, je potrebné prvok pridať do usporiadania.

```
self.inputSelector.addEnabled = False
self.inputSelector.setToolTip("Pick the markup list to be filled")
self.inputSelector.removeEnabled = False
self.inputSelector.setMRMLScene(slicer.mrmlScene)
volumesFormLayout.addRow(self.inputSelectorLabel, self.inputSelector)
```

4.3.3 Implementácia logickej časti modulu

Celá logická časť je spustiteľná tlačidlom Apply. V prvom kroku je potrebné previesť užívateľom zvolené premenné a vstupné dátové štruktúry do premenných, s ktorými bude možné ďalej pracovať. V ukážke je možné vidieť vloženie do premennej štruktúru označených bodov, z tejto štruktúry je vybratý ich počet.

```
fiducialnode = self.markupSelector.currentNode() # markups
pocetfiducials = fiducialnode.GetNumberOfFiducials() # info o markupoch
roinode = self.roiSelector.currentNode() # roi
inputVolume = self.inputSelector.currentNode() # vstup
volumesLogic = slicer.modules.volumes.logic() # pomocna funkcia pre gauss logiku
```

Ďalším krokom je vytvorenie vhodného výstupného obrazového kontajneru. Aj keď užívateľ môže vytvárať obrazové štruktúry, ich parametre sú nevyplnené, a preto je potrebné ich vhodne predvyplniť tak, aby boli rovnaké s vstupnými dátami. Ide hlavne o koordináty a rozlíšenie obrazových dát.

```
outputVolume = volumesLogic.CloneVolume(slicer.mrmlScene, inputVolume,
outputName) # vytvorenie kopie vystupneho suboru z vstupneho
self.outputSelector.setCurrentNode(outputVolume)
ijkToRas = vtk.vtkMatrix4x4() #transofrmacna matica
outputVolume.GetIJKToRASMatrix(ijkToRas)#ziskanie transformacnej matice
imageDataout = outputVolume.GetImageData()# ziskanie obrazovych dat
extent = imageDataout.GetExtent() # ziskanie rozlisenia obrazu
```

Pretože rozšírenie je zamerané na prácu s viacerými vstupnými obrazmi, je vhodné ich previesť do jednej štruktúry, s ktorou bude modul ďalej pracovať. Na ukážke je vloženie jedného dátového kontajnera do štruktúry.

```
inputvol = [] # vytvorenie datovej struktury
inputvol.append(self.inputSelector.currentNode()) # pridanie dat do struktury
```

Po tom ako boli všetky vstupné dáta vložené do štruktúry, na základe užívateľskej voľby sa môže vykonať časť kódu kde sa ku každému vstupnému obrazu vygeneruje jeho ekvivalent upravený funkciou Gaussian Blur.

```
if self.dogauss.checked == 1:  # ak je zatrhnuta volba gauss
  gaussVolume = []
  gausdim = int(self.gaussdimension.value) # velkost radiusu
  gaussLogic = slicer.modules.volumes.logic() # logika gauss
```

```
        gaussVolume.append(gaussLogic.CloneVolume(slicer.mrmlScene,
self.inputSelector.currentNode(), "Gauss1")) # kopia vstupnych dat pre gauss
        gauss = vtk.vtkImageGaussianSmooth() #volba funkcie gauss
        gauss.SetInputData(self.inputSelector.currentNode().GetImageData())
        gauss.SetDimensionality(3) # 3D gauss
         gauss.SetStandardDeviation(gausdim) #nastavenie radiusu
         gauss.Update() # vykonanie funkcie
         ijkToRAS = vtk.vtkMatrix4x4()
         self.inputSelector.currentNode().GetIJKToRASMatrix(ijkToRAS)
         gaussVolume[0].SetIJKToRASMatrix(ijkToRAS) # nastavenie transformacnej mat.
         gaussVolume[0].SetAndObserveImageData(gauss.GetOutput())# ukazka obrazu
inputvol.append(gaussVolume[0]) # pridanie do struktury obrazov
```

Po vytvorení sa tento upravený obraz pridá na koniec štruktúry, ktorá sa ďalej bude upravovať tak, aby ju bolo možné spracovať. Keďže sa jedná o dátovú štruktúru, ktorá zapuzdruje okrem obrazových dát aj viacero iných parametrov, je potrebné hodnoty obrazových bodov vybrať a pracovať s nimi oddelene. Ostatné parametre v spracovaní pomocou SVM nevyužívame.

```
    inputvolimage = []
    for i in range(len(inputvol)):
      inputvol[i].GetIJKToRASMatrix(ijkToRas)
      inputvolimage.append(inputvol[i].GetImageData()) # vyber obrazu z struktury
```

Užívateľ mal možnosť zvoliť pri nastavení parametrov možnosť normalizácie obrazových dát. Predpokladom je nájdenie maxima pre každý vstupný obraz. Vstupné dáta sa prevedú na pole typu numpy, v ktorom je jednoduché nájsť maximum. V MRML dátovej štruktúre je tento krok problematický.

```
    maxvalue = [0] * len(inputvolimage) # struktura pre normalizaciu
    if self.normalize.checked == 1:
      for current in xrange(len(inputvolimage)):
        a =
vtk.util.numpy_support.vtk_to_numpy(inputvolimage[current].GetPointData().GetScal
ars())  # konverzia do numpy pre najdenie maxima
        maxvalue[current] = a.max() # najdenie maxima v numpy array
```

Pre trénovanie modelu SVM je potrebné získať zo vstupných dát trénovací dátový set. Hodnoty sú vyberané zo štruktúry zapuzdrujúcej označovacie dáta. Z týchto dát sa vyberú koordináty, prevedú na koordinačný systém, zhodný s obrazovými

dátami. Na týchto koordinátoch sa nachádza užívateľom zvolený trénovací bod, ktorému bol pridelený klasifikátor pre ďalšie spracovanie. Vytvorenie testovacej štruktúry je základným predpokladom pre trénovanie modelu SVM. Ukážka zobrazuje časť kódu, ktorý vytvára štruktúru pre pozitívne klasifikované trénovacie body.

```
    for i in range(pocetfiducials): # pre kazdy oznaceny bod v markupe
      world = [0, 0, 0, 0]
      fiducialnode.GetNthFiducialWorldCoordinates(i, world) # ziska koordinaty bodu
      ras2ijk = vtk.vtkMatrix4x4()
      inputVolume.GetRASToIJKMatrix(ras2ijk) # ziska ras2ijk maticu
      ijkpoint = [int(round(c)) for c in ras2ijk.MultiplyPoint(world)] # prevedie ras na ijk
     for current in xrange(len(inputvolimage)):
        pixelvalue =
inputvolimage[current].GetScalarComponentAsDouble(int(ijkpoint[0]),
int(ijkpoint[1]),int(ijkpoint[2]), 0)  #ziska hodnoty pix z obrazovych  dat
        s[current + 1] = pixelvalue # vlozenie do trenovacej struktury
        x.append(s)
   y = [1] * len(x) # klasifikator pre trenovacie data (positive)
```

Ďalšou potrebnou časťou, ktorú je potrebné definovať, je určenie parametrov pre model SVM. Ide o výber preddefinovaných hodnôt z užívateľského rozhrania.

Vytvorenú štruktúru je možné ďalej spracovať, a natrénovať model SVM pomocou knižnice libsvm. Po natrénovaní modelu máme všetky potrebné dáta na to, aby sme mohli uskutočniť SVM klasifikáciu na testovacích dátach. Aby nebolo potrebné klasifikovať celý obraz, používame región záujmu. Ten zabezpečí, aby nebola klasifikovaná oblasť, ktorá nás nezaujíma a zároveň tak je potrebné spracovať menšiu časť obrazu, čo má za následok zrýchlenie celého procesu. Problém s dátovou štruktúrou ROI v programe 3D Slicer je v jej zápise koordinátov. ROI má hranatú formu. Zapisuje sa pomocou koordinátov stredu a polomeru k okrajovým bodom. Tieto koordináty sú taktiež v inej sústave, a je potrebné ich previesť na koordináty zhodné s vstupným obrazom. Nasledujúcou časťou je prevedenie koordinátov ROI na hranové súradnice v sústave IJK pre každú hranu ROI. Je to potrebné, z dôvodu vymedzenia vstupných dát do klasifikátoru SVM.

```
    roiCenter = [0, 0, 0]
    roiRadius = [0, 0, 0]
    roinode.GetXYZ(roiCenter) # ziska koordinaty ROI - stredovy bod
    roinode.GetRadiusXYZ(roiRadius) # ziska radius - na kazdu stranu jeden
    roiCorner1 = [roiCenter[0] + roiRadius[0], roiCenter[1] + roiRadius[1], roiCenter[2]
```

```
+ roiRadius[2], 1]  # ukazka vypoctu pre jeden roh, je treba pre kazdy roh kocky
    ras2ijk = vtk.vtkMatrix4x4()
    inputVolume.GetRASToIJKMatrix(ras2ijk)
    roiCorner1ijk = ras2ijk.MultiplyPoint(roiCorner1) # prevedenie koordin. na IJK
    lowerIJK = [0, 0, 0]
    upperIJK = [0, 0, 0]
# ukazka urcenia zaciatku a konca jednej hrany ROI - celkovo 6 koordinatov, kazda
strana 2x zaciatok a koniec
    lowerIJK[0] = min(roiCorner1ijk[0], roiCorner2ijk[0], roiCorner3ijk[0],
roiCorner4ijk[0], roiCorner5ijk[0], roiCorner6ijk[0], roiCorner7ijk[0], roiCorner8ijk[0])
    upperIJK[0] = max(roiCorner1ijk[0], roiCorner2ijk[0], roiCorner3ijk[0],
roiCorner4ijk[0], roiCorner5ijk[0], roiCorner6ijk[0], roiCorner7ijk[0], roiCorner8ijk[0])
```

Po zistení rozsahu, ktorý je potrebné spracovať sa vytvorí dátová štruktúra potrebná pre klasifikáciu SVM. Princíp je rovnaký ako pri získavaní trénovacej štruktúry. Získajú sa obrazové dáta zo všetkých vstupných obrazov v rozsahu ROI.

```
    for k in xrange(int(lowerIJK[2]), int(upperIJK[2]) + 1):
      for j in xrange(int(lowerIJK[1]), int(upperIJK[1]) + 1):
        for i in xrange(int(lowerIJK[0]), int(upperIJK[0]) + 1): # pre kazdy pixel v ROI
          s = {}
          for current in xrange(len(inputvolimage)): # pre kazdy bod v strukture
            pixelvalue = inputvolimage[current].GetScalarComponentAsDouble(int(i),
int(j), int(k), 0) # vyber bodu do premennej
            if self.normalize.checked == 1:
              pixelvalue = (pixelvalue / float(maxvalue[current])) * 255 # normalizacia
            s[current + 1] = pixelvalue
          x.append(s) # vlozenie do struktury
      print(s)
    y = [-1] * len(x) # testovaci klasifikator - nevyuziva sa
```

Vytvorená dátová štruktúra sa spolu s natrénovaným modelom použije na klasifikáciu testovacích dát. Výstupom je klasifikovaný set hodnôt, ktorý je potrebné uložiť do predvytvoreného výstupného MRML kontajneru.

```
    line = 0
    for k in xrange(int(lowerIJK[2]), int(upperIJK[2]) + 1):
      for j in xrange(int(lowerIJK[1]), int(upperIJK[1]) + 1):
        for i in xrange(int(lowerIJK[0]), int(upperIJK[0]) + 1): #pre vsetky pixely v ROI
```

```
            functionValue = p_labels[line] # vyber s klasifikovanych dat
            imageDataout.SetScalarComponentFromDouble(i, j, k, 0, float(functionValue
))  # vlozenie klasifikovaneho bodu na jeho povodne miesto
            line = line + 1
    imageDataout.Modified() # refresh obrazu, znovunacitanie do GUI
```

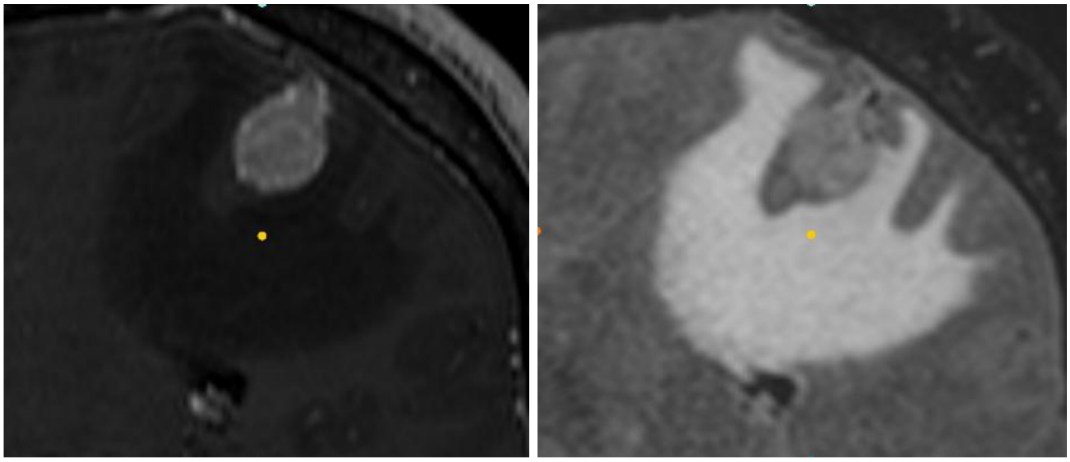
Po uložení do výstupného obrazu, sa tento obraz prevedie ako aktívny na plochu programu 3D Slicer, kde je ho ďalej možné analyzovať a spracovať pomocou ďalších nástrojov. Na záver sa vykoná premazanie nepotrebných MRML obrazových štruktúr, hlavne obrazy z funkcie Gaussian Blur, ktoré zaberajú výrazné množstvo pamäti RAM.

5. VÝSLEDKY NAVRHNUTÉHO RIEŠENIA

V tejto kapitole sú prezentované výsledky navrhnutého riešenia ako sú výsledky segmentácie konkrétnych obrazov či porovnanie s segmentáciou z niektorých jedno parametrických metód ktoré boli spomenuté v kapitole 2. V kapitole je porovnanie výsledkov manuálnej segmentácie vybraných rezov obrazu s automatickým spracovaním.

5.1 POROVNANIE S JEDNOPARAMETRICKÝMI METÓDAMI

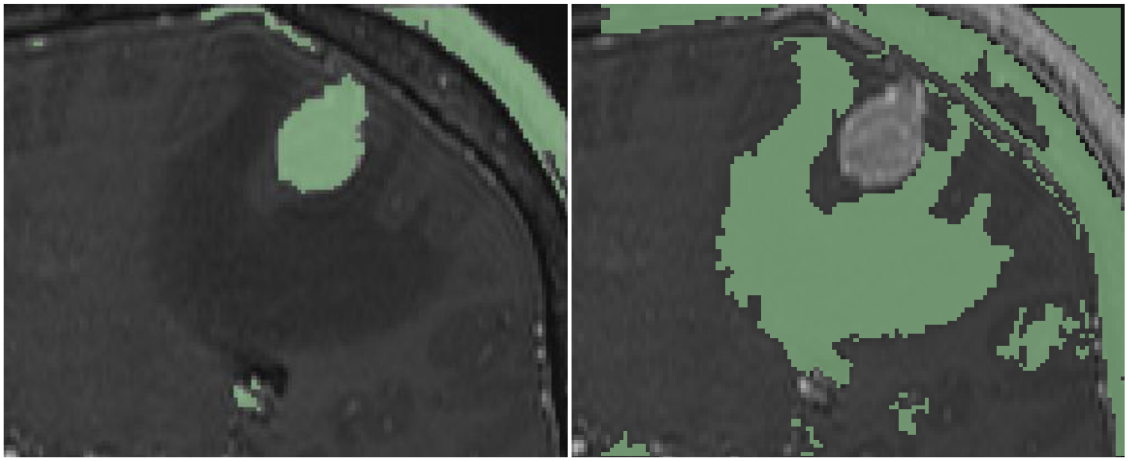
Výsledky segmentácie MR obrazov je možné porovnať s inými metódami. Práca je zameraná na využitie viac parametrického spôsobu segmentácie obrazov. Na obrázku 5.1 je vidieť ukážku vstupných obrazov na ktorých bude vykonávaná porovnanie. Obrazy sú výsekom regiónu záujmu z jedného MR rezu.



Obrázok 5.1: Vstupné obrazy typu T_1 naľavo a T_2 napravo

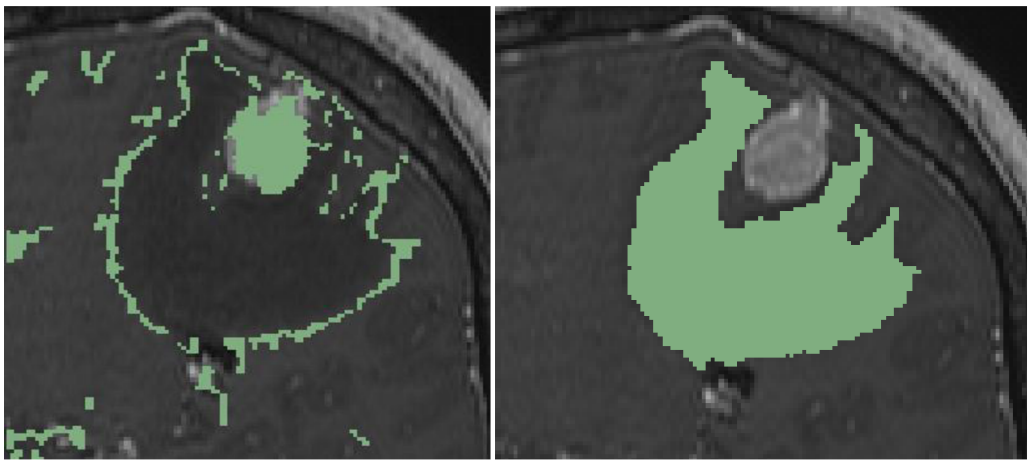
Klasifikačná metóda segmentácie pomocou SVM vyžaduje pre správnu klasifikáciu mať dostatočné množstvo vstupných parametrov. Pri nedostatku týchto parametrov metóda nedosahuje dostatočne kvalitné segmentačné výsledky. Pri použití jedného parametru - jedného vstupného obrazu, je výsledok porovnateľný s inými jedno parametrickými segmentačnými technikami ako je napríklad metóda prahovania alebo metóda rozvodia. Pretože MR obrazy majú špecifiká ako je zvýšený šum prípadne veľmi malé rozdiely intenzít medzi dvoma objektmi, tieto jednoduché metódy zlyhávajú a vykazujú nedostatočne kvalitné výsledky.

Na obrázku 5.2 je vidieť výsledok segmentácie tumoru (naľavo) a edému (napravo), pri použití metódy SVM a jedného vstupného parametru - obrazu typu T_1. Tumor je na tomto obraze zreteľnejší a preto dosahuje lepších segmentačných výsledkov než edém ktorý je v obraze typu T_1 slabo viditeľný.



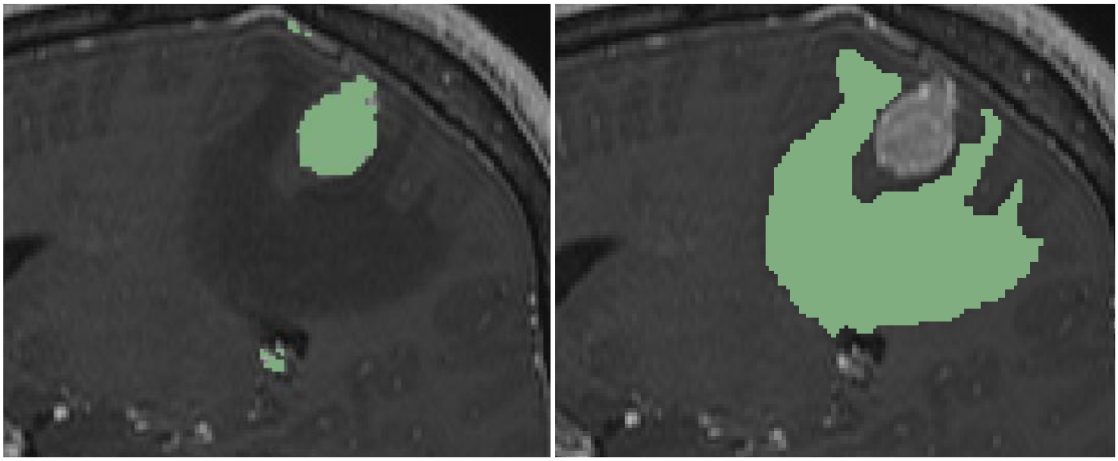
Obrázok 5.2: Výsledok segmentácie SVM v obraze T_1, tumor vľavo, edém vpravo.

Výrazným problémom je aj podobnosť intenzity tumoru alebo edému s inými tkanivami v mozgovej štruktúre. Z toho dôvodu je výsledok segmentácie T_1 (obrázok 5.2) a obrazu T_2 (obrázok 5.3) zaťažený chybou. Výnimkou je segmentácia edému v obraze T_2 kde intenzita edému sa výrazne líši od ostatných tkanív. V tomto prípade je výsledok segmentácie dostatočne úspešný.



Obrázok 5.3: Výsledok segmentácie SVM v obraze T_2, tumor vľavo, edém vpravo.

Segmentácia tumoru metódou SVM v oboch obrazoch T_1 aj T_2 je ale nedostatočná a bolo by potrebné segmentovaný obraz ďalej spracovávať pre získanie dostatočne kvalitného výsledku. Pri použití kombinácie obrazov dosiahneme výrazne lepší výsledok viď obrázok 5.4. Segmentácia tumoru aj edému je zobrazená bez akýchkoľvek dodatočných úprav. Je vidieť že chybná segmentácia je minimalizovaná a takmer žiadne iné tkanivá nie sú obsiahnuté v segmentácií. Môžeme povedať že kombináciou dvoch vstupných obrazov v segmentácii pomocou SVM sme dosiahli výrazne lepší výsledok ako pri oddelenej segmentácii.



Obrázok 5.4: Výsledok segmentácie SVM s použitím obrazov T_1 a T_2, tumor vľavo, edém vpravo.

Z testovania vyplýva, že použitie viacerých parametrov na segmentovanie obrazu metódou SVM je výrazne lepšie než použitie jedného parametru.

Parametre segmentácie SVM použité v testovaní (parametre nastaviteľné v programe 3D Slicer):

- Gaussovské jadro,
- *Gamma in kernel* = 10^{-6} (parameter γ).

Matematický model segmentačnej metódy:

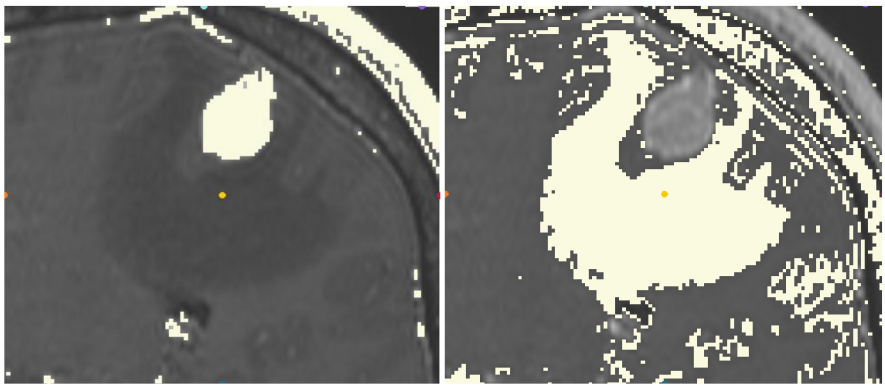
$$K(x, y) = e^{(-10^{-6}\,||x-y||^2)}. \tag{5.1}$$

V tabuľke je zobrazená presnosť klasifikácie SVM pre rôzne vstupné obrazy a DICE koeficient vzhľadom k referenčným segmentom vytvoreným empiricky.

	Segmentácia T_1 obrazu		**Segmentácia T_2 obrazu**		**Segmentácia T_1 a T_2 obrazu**	
	Tumor	**Edém**	**Tumor**	**Edém**	**Tumor**	**Edém**
Presnosť klasifikácie SVM [%]	95,04	79,01	89,11	98,74	99,17	98,69
DICE koeficient [-]	0,52	0,63	0,30	0,94	0,86	0,97

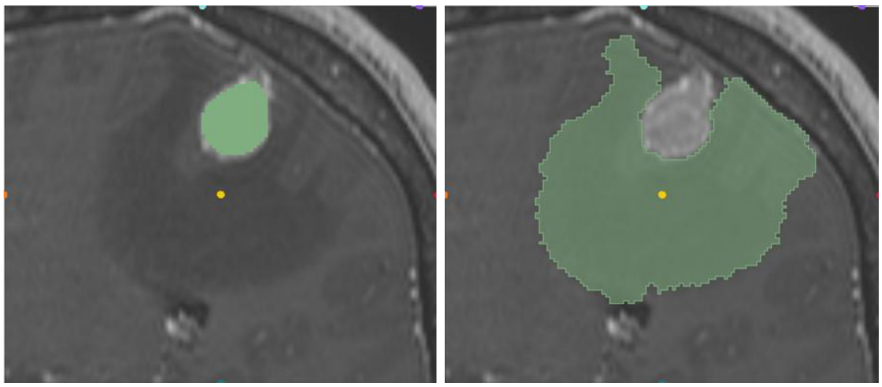
Tabuľka 3: Presnosť klasifikácie SVM vzhľadom k počtu parametrov

Program 3D Slicer obsahuje štandardne niektoré jednoduché segmentačné metódy. Jedným zo základných segmentačných techník je metóda prahovania. Na obrázku 5.5 je vidieť výsledok segmentácie pomocou tejto metódy. Úroveň prahu bola určená empiricky. Pretože sa jedná o jednoparametrickú metódu, výsledná segmentácia tumoru aj edému obsahuje výrazne množstvo nesprávne segmentovaných tkanív. V prípade tumoru ide o kostené štruktúry. V prípade edému ide o akékoľvek mozgové tekutiny ktoré vykazujú podobné intenzity ako edém. Presnosť segmentácie metódou prahovania podľa DICE koeficientu bola určená ako 0,55 pre segmentáciu tumoru a 0,65 pre segmentáciu edému.



Obrázok 5.5: Výsledok segmentácie metódou prahovania v obraze typu T_1, prah určený empiricky. Tumor vľavo, edém vpravo.

Na obrázku 5.6 je zobrazený výsledok segmentácie pomocou metódy rozvodia. V programe 3D Slicer bolo v prvom kroku vyznačené orientačne ktoré štruktúry patria do oblasti ktorá nás zaujíma a ktoré nie. Na základe týchto vstupných údajov, funkcia vysegmentuje podobné oblasti. Dosahuje lepšie výsledky segmentácie ako metóda prahovania. Pretože ide stále o jedno parametrickú metódu dochádza k segmentácii k výrazným odchýlkam. Presnosť segmentácie metódou rozvodia podľa DICE koeficientu pre testovací obraz bola určená ako 0,82 pre tumor a 0,76 pre edém.



Obrázok 5.6: Výsledok segmentácie metódou rozvodia v obraze typu T_1, tumor vľavo, edém vpravo.

5.2 TESTOVANIE NA REÁLNYCH SUBJEKTOCH

Výsledky segmentácie pomocou metódy SVM závisia aj od kvality vstupných obrazov. V medicínskych obrazoch, obzvlášť pri MR skenoch rôznych osôb závisí kvalita ďalšieho spracovania na mnohých faktoroch. Kvalita MR obrazu je závislá od podmienok pri tvorbe obrazu, v prípade že osoba v prístroji pohne dochádza k rozmazaniu prípadne k zašumeniu obrazu a odrazom. Taktiež výsledok spracovania závisí aj od typu skúmaného objektu, tumoru či edému. Sú prípady keď má tumor metastázy a je rozšírený do veľkej časti mozgu. v prípade že sú tieto metastázy malé je problematické ich vôbec odhaliť. Je to spôsobené aj nízkou rozlišovacou schopnosťou MR technológie. V tejto kapitole sú zobrazené výsledky segmentácie metódou SVM a porovnané s manuálnou segmentáciou. Manuálna segmentácia bola vytvorená dopredu podľa informácii od medicínskeho špecialistu. Pretože manuálna segmentácia nie je dostatočne dokonalá hlavne v nehomogénnych skúmaných objektoch, dochádza k výraznejším odchýlkam v obrazoch obsahujúcich tieto objekty. Manuálne spracovanie nie je schopné vyznačiť všetky nehomogenity v obraze, jedná sa o približné určenie.

Parametre segmentácie SVM použité v testovaní (parametre nastaviteľné v programe 3D Slicer):

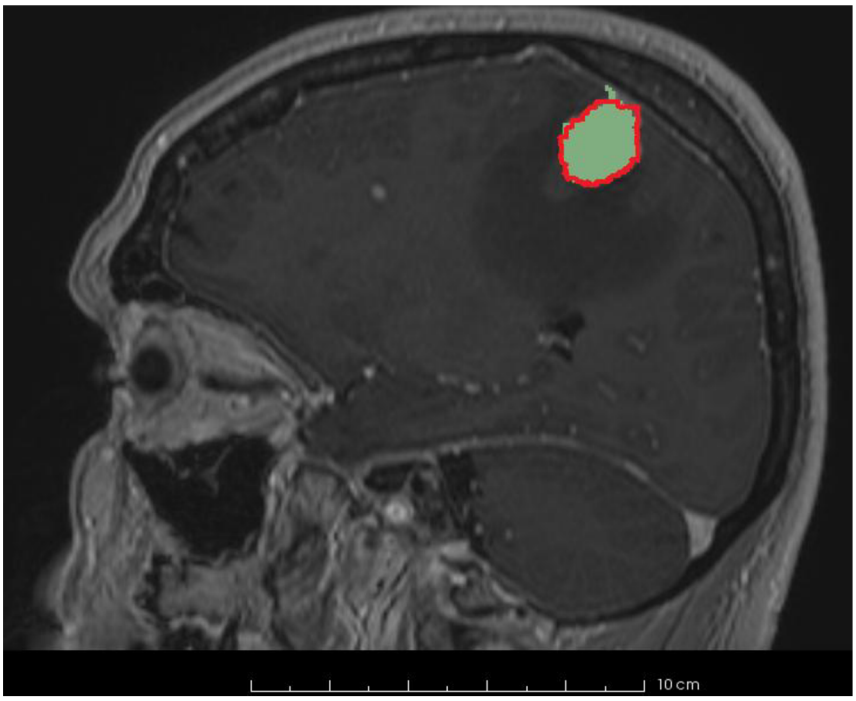
- Gaussovské jadro,
- *Gamma in kernel* = 10^{-6} (parameter γ),
- *Gauss filter enabled,*
- *Gauss radius* = 2 (veľkosť jadra je 5 × 5 × 5).

Matematický model segmentačnej metódy:

$$K(x, y) = e^{(-10^{-6}\,||x-y||^2)}. \tag{5.2}$$

Ako vstupné obrazy do segmentácie boli použité štruktúrne obrazy T_1 a T_2. Z dôvodu dosiahnutia presnejších výsledkov je použitá funkcia Gaussian filter u oboch obrazov. Táto funkcia vytvorí ďalšie vyhladené ekvivalentné obrazy k vstupným obrazom. Do klasifikátora SVM teda vstupujú 4 parametre, dva štruktúrne obrazy a dva filtrované štruktúrne obrazy.

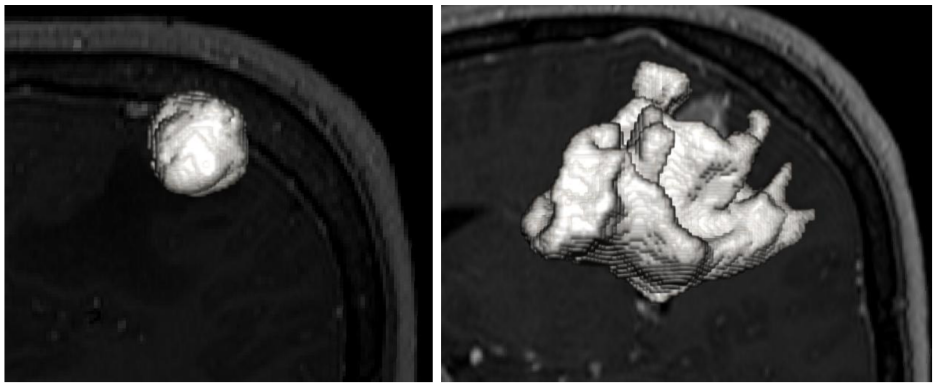
Na obrázku 5.7 je zobrazený výsledok segmentácie tumoru. Zelenou farbou je podfarbená segmentácia vytvorená manuálne. Červenou farbou je vyznačená hranica segmentácie vytvorenej automaticky metódou SVM s vyššie uvedenými parametrami. Na obrázku 5.8 je zobrazený výsledok segmentácie edému. Rovnako ako pri predchádzajúcom obraze bolo použité zhodné farebné označenie.



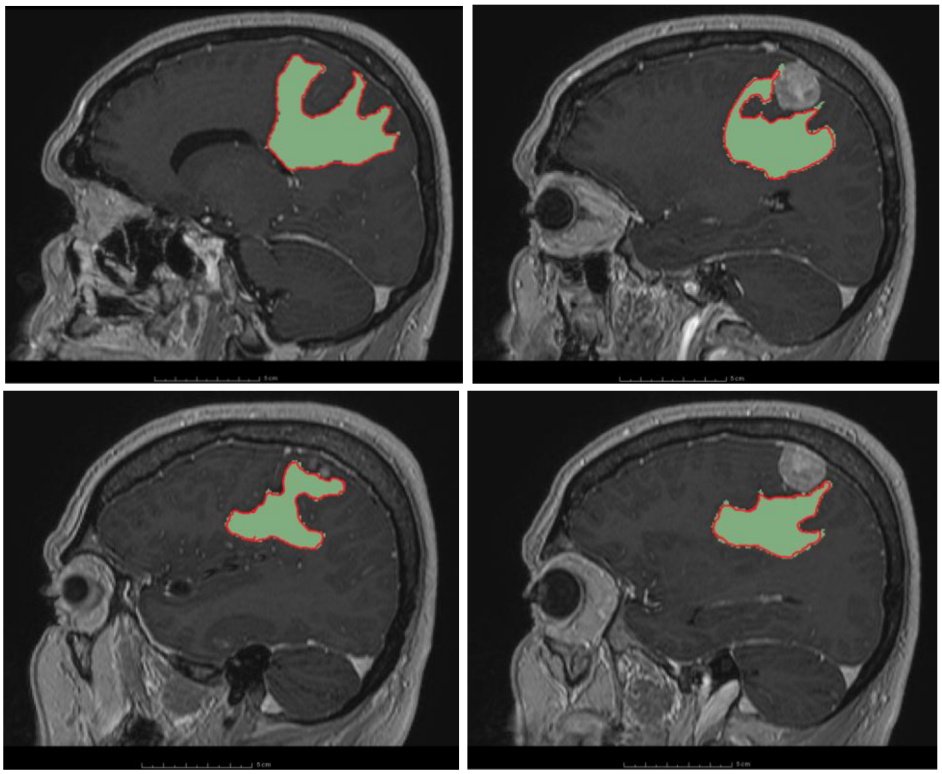
Obrázok 5.7: Segmentácia tumoru, zelená - referenčná, červená - SVM segmentácia

Obrázok 5.8: Segmentácia edému, zelená - referenčná, červená - SVM segmentácia

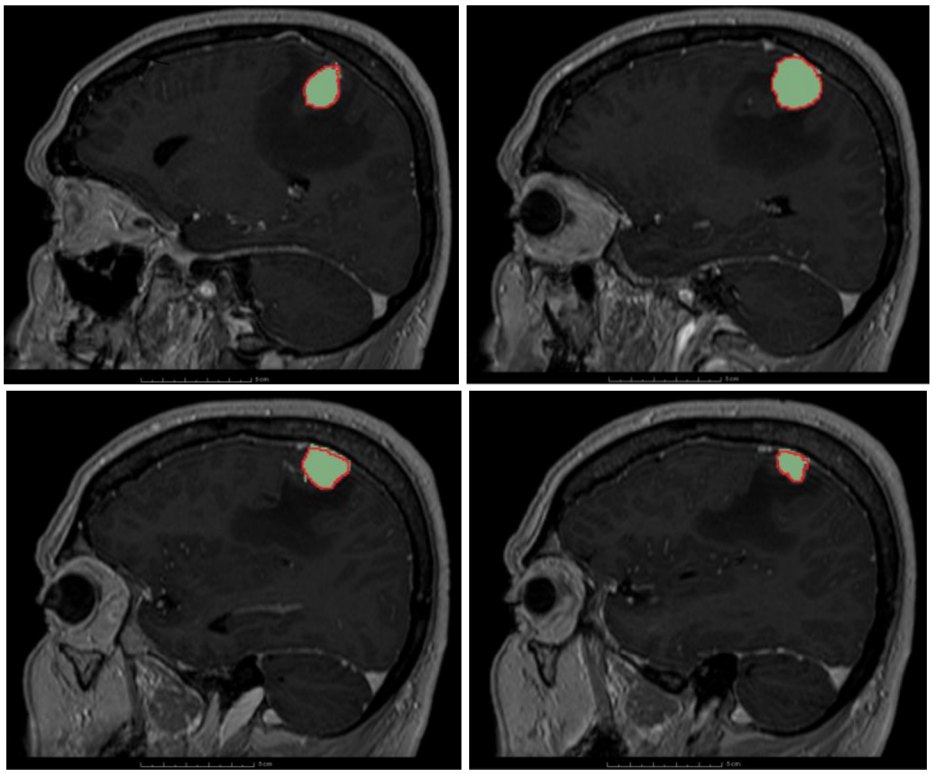
Obrázok 5.9 zobrazuje ukážku 3D štruktúr edému a tumoru u vzorového subjektu. Obrázky 5.10 a 5.11 zobrazujú niektoré ďalšie rezy rovnakým testovacím subjektom. V tabuľke 4 je zobrazené porovnanie presnosti segmentácie vzhľadom k manuálnemu spracovaniu.



Obrázok 5.9: 3D štruktúra tumoru vľavo, 3D štruktúra edému vpravo



Obrázok 5.10: Vybrané rezy MR snímkov hlavy testovacieho subjektu 1 s výsledkom segmentácie edému. Zelená - referenčná, červená - SVM segmentácia



Obrázok 5.11: Vybrané rezy MR snímkov hlavy testovacieho subjektu 1 s výsledkom segmentácie tumoru. Zelená - referenčná, červená - SVM segmentácia

	Segmentácia tumoru		**Segmentácia edému**	
	Presnosť klasifikácie SVM [%]	**DICE koeficient [-]**	**Presnosť klasifikácie SVM [%]**	**DICE koeficient [-]**
Rez 1	99,48	0,93	99,04	0,98
Rez 2	99,60	0,93	98,78	0,97
Rez 3	99,79	0,98	98,89	0,96
Rez 4	99,56	0,93	99,89	0,96
Rez 5	99,66	0,89	99,47	0,97

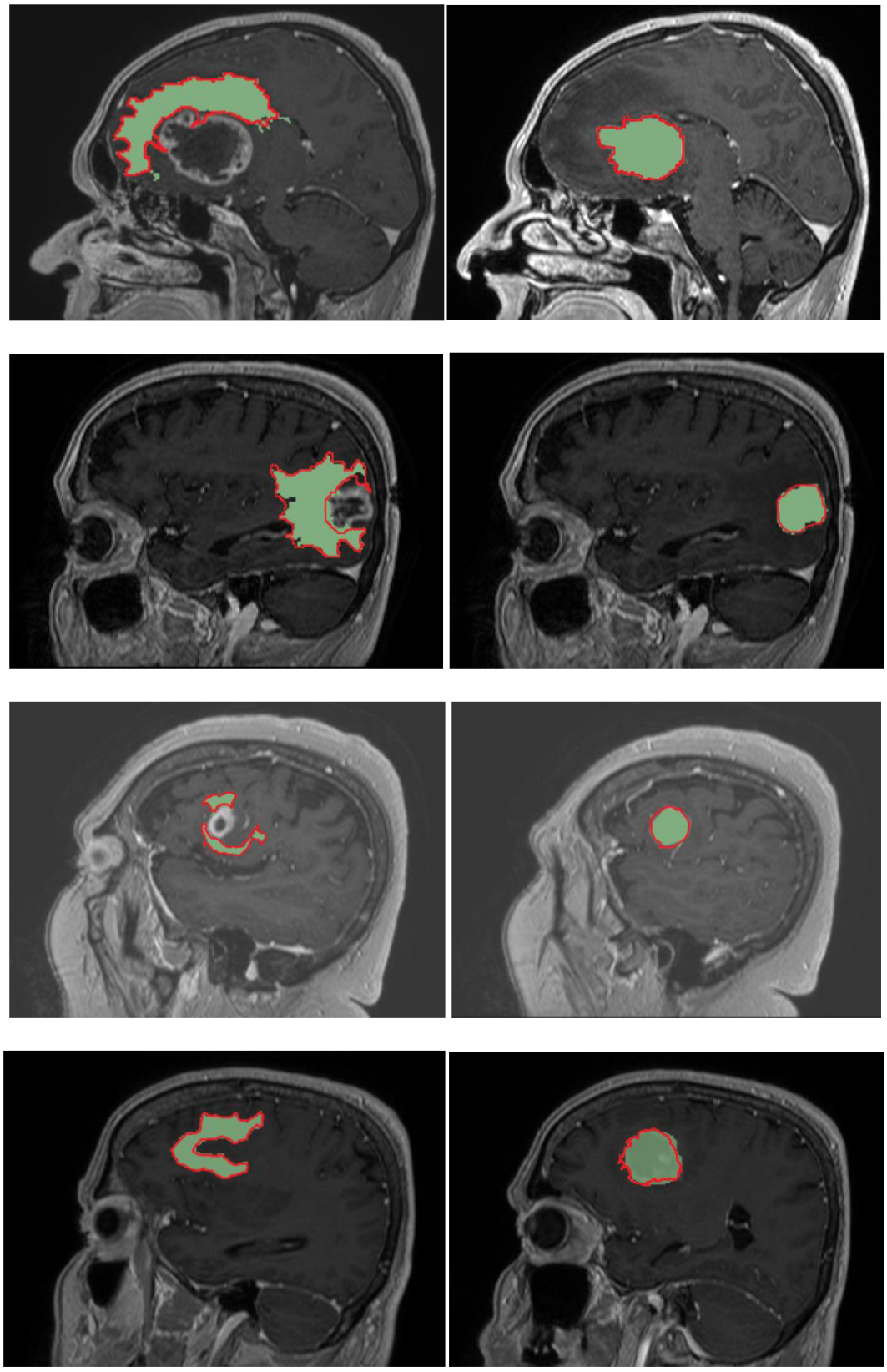
Tabuľka 4: Presnosť klasifikácie SVM na rôznych rezoch rovnakého subjektu

Celková presnosť segmentácie pomocou metódy SVM bola určená ako aritmetický priemer výsledkov presnosti klasifikácie SVM a DICE koeficientu u niekoľkých skúmaných subjektov. V tabuľke 5 sú zobrazené výsledky presnosti segmentácie na testovacích subjektoch. Výpočet pre jeden subjekt bol stanovený ako priemer 10 náhodných rezov MR skenom ktoré obsahovali skúmaný tumor alebo edém.

	Segmentácia tumoru		**Segmentácia edému**	
	Presnosť klasifikácie SVM [%]	**DICE koeficient [-]**	**Presnosť klasifikácie SVM [%]**	**DICE koeficient [-]**
Subjekt 1	99,62 ± 0,12	0,93 ± 0,03	99,02 ± 0,27	0,96 ± 0,01
Subjekt 2	97,76 ± 1,34	0,86 ± 0,04	86,18 ± 1,89	0,82 ± 0,06
Subjekt 3	98,74 ± 0,25	0,85 ± 0,03	96,06 ± 0,88	0,86 ± 0,02
Subjekt 4	98,87 ± 0,63	0,88 ± 0,07	98,63 ± 0,36	0,82 ± 0,07
Subjekt 5	96,64 ± 1,24	0,88 ± 0,01	98,25 ± 0,87	0,82 ± 0,19
Priemer	**98,33 ± 0,72**	**0,88 ± 0,04**	**95,16 ± 1,08**	**0,86 ± 0,07**

Tabuľka 5: Výsledky segmentácie edému a tumoru na viacerých testovacích subjektoch

Výsledky segmentácie boli zámerne rozdelené do dvoch kategórii. Je to z dôvodu, že tumor v skúmaných subjektoch má pomerne často homogénny tvar a preto bolo jednoduchšie vyznačiť ručne referenčný tvar. V segmentácii edému bola tvorba referenčných segmentov obtiažná a dochádzalo k nepresnostiam. Z toho dôvodu je výsledná presnosť segmentácie v prípade edému horšia. Na obrázku 5.12 je vidieť ukážkové náhľady na výslednú segmentáciu vybraných rezov z testovaných subjektov.



Obrázok 5.12: Ukážka segmentácie tumoru a edému na subjektoch 2 až 5, edém vľavo, tumor vpravo. Zelené podfarbenie reprezentuje referenčnú segmentáciu, červený obrys SVM segmentáciu

ZÁVER

Túto prácu som venoval skúmaniu a implementácii viac parametrickej metódy SVM na segmentáciu obrazov, ktoré vznikajú pri tomografickom vyšetrení, do programu 3D Slicer. Pretože obrazové dáta v medicínskom prostredí sú veľmi previazané, napríklad 3D obraz mozgu s viacerými typmi snímkov, je pri následnej analýze dôležité správne tieto informácie zaradiť. Pre personál rozhodujúci o následnej diagnóze je dôležité dostať vždy čo najpresnejšie vyhodnotenie.

Práca sa zaoberá vytvorením rozširovacieho modulu do programu 3D Slicer, ktorý využíva funkciu metódy SVM. Je to z toho dôvodu, že software 3D Slicer je optimalizovaný na využitie v experimentálnom prostredí a jeho modulárna architektúra umožňuje vytvárať rozšírenia a zároveň využívať už podporované funkcie. S využitím rozšírení do programu sme schopní vytvoriť optimálnu funkciu zameranú na segmentáciu a vyznačenie konkrétnej oblasti záujmu v ľudskom tele. Takýmto spôsobom je možné dosiahnuť ucelený náhľad v 3D a umožniť tak doktorom optimalizovať ďalšie kroky v liečení.

V práci je taktiež obsiahnutá aj základná analýza tvorby rozšírení do programu 3D Slicer s ukážkami zdrojových kódov pre univerzálne použitie. Táto problematika je dôkladne popísaná v kapitole 4. Tento úvod do problematiky vývoja rozširovacích modulov v programe 3D Slicer môže výrazne uľahčiť vznik budúcim experimentálnym rozšíreniam založeným na platforme modulov písaných v jazyku Python.

Hlavným cieľom práce bolo vytvorenie kompletného rozširovacieho modulu programu 3D Slicer. Tento modul plne implementuje funkcie klasifikačných schopností SVM a využíva ich v spracovaní medicínskych obrazov. Grafické rozhranie umožňuje nastaviteľný počet vstupných obrazov až na maximum 10 obrazov. Pri použití Gaussiánovho filtra sa z týchto obrazov vytvoria ich filtrované ekvivalenty čo zlepšuje následný segmentačný výsledok. Rozhranie implementuje nastavenie všetkých základných parametrov SVM metódy vrátanie voľby jadier a ich vnútorných koeficientov. Rozširovací modul vytvorený pre tento software implementuje dva spôsoby využitia. Jedným je segmentácia obrazu s využitím už klasifikovaných dát v rovnakom obraze. Tento spôsob je využiteľný pri znalosti niektorých pozitívnych a negatívnych bodov z jedného rezu alebo celého 3D skenu. Informácie sa využijú na trénovanie SVM klasifikátora a následne aplikujú na segmentovanie ostatných rezov. Druhým využitím je možnosť vytvorenia veľkej trénovacej databázy, napríklad z viacerých známych subjektov a tento model využiť na segmentáciu neznámeho subjektu bez akýchkoľvek dodatočných znalostí pri porovnateľných vlastnostiach trénovacích a testovacích dát.

SVM je vhodná segmentácia na zašumené prostredie, prípadne inak poškodené obrazové informácie a je ideálna na analýzu obrazu pomocou viacerých vstupných obrazov. Táto metóda umožňuje vopred natrénovať na obraze, ktoré oblasti patria do regiónu záujmu a ktoré nie. Metóda zlyháva pri použití nedostatočného množstva parametrov. V prípade mozgových tkanív dosahuje nízku kvalitu segmentácie podľa DICE koeficientu. V prípade skúmania tumoru jedným rezom je výsledný koeficient 0,52 v obraze typu T_1 a 0,30 v obraze typu T_2. V prípade skúmania edému v jednom testovacom reze je výsledný DICE koeficient rovný 0,63 v obraze typu T_1 a 0,94 v obraze typu T_2. Pre porovnanie, rovnaký obrazový snímok bol segmentovaný jednoduchými metódami, ktoré program 3D Slicer štandardne obsahuje. Pri použití metódy prahovania sme dosiahli presnosť segmentácie podľa DICE koeficientu 0,55 pre segmentáciu tumoru a 0,65 pre segmentáciu edému. Ďalšou metódou bola segmentácia metódou rozvodia. Pre daný testovací obraz bol DICE koeficient určený hodnotou 0,82 pre tumor a 0,76 pre edém. Výstup potvrdil, že segmentácia pomocou SVM je nedostatočná pri použití len jedného parametru. Výrazné zlepšenie nastáva pri použití viacerých dostupných obrazov a ich upravených ekvivalentov. Pri použití kombinácie obrazov T_1 a T_2 metóda SVM dosahuje u testovacieho obrazu presnosť podľa DICE koeficientu 0,86 pre tumor a 0,97 pre edém. V prípade edému je použitie SVM o dvoch parametroch v segmentácii dostatočné, čo potvrdzuje zistený koeficient. Pre segmentáciu tumoru nie je výsledok úplne kvalitný, a preto bol zvýšený počet vstupných parametrov o filtrované ekvivalenty vstupných obrazov. DICE koeficient pre rovnaký testovací rez pri použití metódy SVM o štyroch vstupných parametroch dosahuje hodnoty 0,93 pre segmentáciu tumoru a 0,98 pre segmentáciu edému. Testovanie potvrdzuje, že metóda SVM zvyšuje svoju presnosť s narastajúcim počtom parametrov.

Ďalšie testovanie bolo zamerané na otestovanie viacerých testovacích subjektov s rôznymi typmi tumorov. Toto testovanie malo za úlohu zistiť experimentálne celkovú kvalitu segmentácie SVM. Z každého testovacieho subjektu bolo použité pre získanie výpočtu 10 rezov obsahujúcich skúmaný tumor či edém. Výsledkom je pre segmentáciu tumoru hodnota DICE koeficientu $0,88 \pm 0,04$ a pre štruktúru edému bol stanovený DICE koeficient $0,86 \pm 0,07$. Výpočet bol zámerne rozdelený na skúmanie tumoru a edému z dôvodu rozličných vlastností a zväčšenej nehomogenite pri určovaní referenčných hraníc edému. Testovacia segmentácia bola vykonávaná na 4 vstupných parametroch. Presnosť segmentácie metódou SVM je možné ďalej zvýšiť využitím pomocných filtračných mechanizmov, ktoré dokážu získať dôležité informácie aj z obrazu nedostatočnej kvality.

LITERATÚRA

[1] MILDENBERGER, P. et al. *Introduction to the DICOM standard*. In European Radiology, 2001, roč. 12, č. 4, s. 1 – 21.

[2] ABE, S. *Support vector machines for pattern classification*. London : Springer, 2005. 343 s. ISBN 18-523-3929-2.

[3] HOLČÍK, J. *Analýza a klasifikace dat*. Brno: Akademické nakladatelství CERM, 2012. 111 s. ISBN 978-807-2047-932.

[4] JIMÉNEZ, O. –MÜLLER, H. *Prototype of 3D annotation software*. 2013. [Online]. [Cit. 2014-11-30]. Dostupné na internete: http://www.visceral.eu/assets/Uploads/Deliverables/VISCERAL-D1.1.pdf

[5] *3D Slicer*. [Online]. [Cit. 2014-11-30]. Dostupné na internete: http://www.slicer.org

[6] FEDOROV, A. et al. *3D Slicer as an image computing platform for the Quantitative Imaging Network*. In Magnetic Resonance Imaging, 2012. s. 1323 – 1341. ISSN 0730725x. Dostupné na internete: http://linkinghub.elsevier.com/retrieve/pii/S0730725X12001816

[7] CHANG, C.; LIN, C. *A library for support vector machines.* In ACM Transactions on Intelligent Systems and Technology, 2011, vyd. 2.[Online]. [Cit. 2014-11-30]. Dostupné na internete: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[8] HLAVÁČ, V.; ŠONKA, M.: *Počítačové vidění*. Praha : Grada, 1992. 272 s. ISBN 80-854-2467-3.

[9] GONZALEZ, R.; WOODS, R.: *Digital image processing*. Pearson : Upper Saddle River, 2008. 954 s. ISBN 01-316-8728-X.

[10] DUDA, R. *Pattern classification*. New York: J. Wiley, 2001. 654 s. ISBN 04-710-5669-3.

[11] STEINWART, I.; CHRISTMANN, A.: *Support vector machines*. New York : Springer, 2008. 601 s. ISBN 9780387772424.

[12] NELLO, C. *Support vector machines*. New York : Cambridge Press, 2000. 189 s. ISBN 05-217-8019-5.

[13] CORTES, C.; VAPNIK, V. *Machine Learning*. 1995. s. 273 – 297. ISSN 1573-0565. [Online]. [cit. 2014-11-30]. Dostupné na internete: http://link.springer.com/10.1023/A:1022627411411

[14] BURGES, C. *A Tutorial on Support Vector Machines for Pattern Recognition*. In Data Mining and Knowledge Discovery, 1998. s. 121 – 167. [Online]. [cit. 2014-11-30]. Dostupné na internete: http://research.microsoft.com/pubs/67119/svmtutorial.pdf