VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2017 Bc. Daniel Chalupa

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ
DEPARTMENT OF BIOMEDICAL ENGINEERING

ROZŠIŘUJÍCÍ MODUL PLATFORMY 3D SLICER PRO SEGMENTACI TOMOGRAFICKÝCH OBRAZŮ
3D SLICER EXTENSION FOR TOMOGRAPHIC IMAGES SEGMENTATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Daniel Chalupa
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Jan Mikulka, Ph.D.
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Biomedicínské inženýrství a bioinformatika**
Ústav biomedicínského inženýrství

Student: Bc. Daniel Chalupa ***ID:*** 155578
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Rozšiřující modul platformy 3D Slicer pro segmentaci tomografických obrazů

POKYNY PRO VYPRACOVÁNÍ:

1) Proveďte literární rešerši v oblasti segmentačních metod vhodných pro zpracování tomografických dat. 2) Navrhněte optimální metodiku automatizovaného zpracování tomografických dat pomocí algoritmů strojového učení. 3) Prostudujte a popište možnosti rozšíření platformy 3D Slicer pro zpracování trojrozměrných dat. 4) Navrhněte vhodný způsob implementace rozšiřujícího modulu pro trojrozměrnou segmentaci tomografických obrazů. 5) Proveďte implementaci zvoleného algoritmu pro segmentaci tomografických obrazů v jazyce C++ nebo Python. 6) Na volně dostupných obrazových databázích testujte nový modul a proveďte analýzu kvality segmentace. 7) Proveďte diskuzi získaných výsledků a zhodnoťte účinnosti a využitelnost řešení.

DOPORUČENÁ LITERATURA:

[1] DUDA, R.O., HART, P.E., STORK, D.G. Pattern Classification. Wiley, 2001.

[2] ŠONKA, M., HLAVÁČ, V., BOYLE, R. Image Processing, Analysis, and Machine Vision. Thomson, 2008.

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 19.5.2017

Vedoucí práce: doc. Ing. Jan Mikulka, Ph.D.
Konzultant:

prof. Ing. Ivo Provazník, Ph.D.
předseda oborové rady

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

ABSTRAKT

Tato práce pojednává o využití strojového učení při úlohách klasifikace medicínských obrazů. Obsahuje literární rešerši pojednávající o klasických a moderních metodách segmentace obrazů. Hlavním cílem práce je navržení a vytvoření rozšíření pro platformu 3D Slicer. Rozšíření využívá strojové učení ke klasifikaci obrazů dle zadaných parametrů. Testování rozšíření probíhá na reálných tomografických obrazech z nukleární magnetické rezonance a sleduje přesnost klasifikace a využitelnost v praxi.

KLÍČOVÁ SLOVA

3D Slicer, C++, klasifikace, modul, náhodný les, segmentace, strojové učení, support vector machine, rozšíření

ABSTRACT

This work explores machine learning as a tool for medical images' classification. A literary research is contained concerning both classical and modern approaches to image segmentation. The main purpose of this work is to design and implement an extension for the 3D Slicer platform. The extension uses machine learning to classify images using set parameters. The extension is tested on tomographic images obtained by nuclear magnetic resonance and observes the accuracy of the classification and usability in practice.

KEYWORDS

3D Slicer, C++, classification, extension, machine learning, module, segmentation, support vector machine, random forest

CHALUPA, Daniel. *Rozšiřující modul platformy 3D Slicer pro segmentaci tomografických obrazů*. Brno, 2017, 60 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínského inženýrství. Vedoucí práce: doc. Ing. Jan Mikulka, PhD.

Vysázeno pomocí balíčku `thesis` verze 2.63; `http://latex.feec.vutbr.cz`

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Rozšiřující modul platformy 3D Slicer pro segmentaci tomografických obrazů“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Janu Mikulkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

OBSAH

Úvod **10**

1 Segmentace tomografických dat **11**
1.1 Předzpracování 11
1.2 Segmentační algoritmy 14

2 Strojové učení **17**
2.1 Umělé neuronové sítě 17
2.2 Support Vector Machines 18
2.3 Náhodný les 20
2.4 Learning Classifier System 21
2.5 Volně dostupné knihovny 22

3 Platforma 3D Slicer **23**
3.1 Knihovny užívané platformou 23
3.2 Možnosti rozšíření 24
3.2.1 Skriptovaná rozšíření 24
3.2.2 Rozšíření s rozhraním přes příkazovou řádku 24
3.2.3 Načitatelná rozšíření 24

4 Vývoj a použití rozšíření **25**
4.1 Stanovené cíle 25
4.2 Základní struktura kódu 26
4.3 Práce se soubory 28
4.4 Předzpracovací algoritmy 29
4.5 Klasifikátory 29
4.6 Uživatelská příručka 31

5 Dosažené výsledky **35**
5.1 C SVM - Dlib 36
5.2 Nu SVM - Dlib 39
5.3 Náhodný les - Shark 43
5.4 Srovnání klasifikátorů 46

6 Diskuze **48**

Závěr **49**

Literatura **51**

Seznam symbolů, veličin a zkratek **55**

Seznam příloh **56**

A Nastavení vývojového prostředí pro rozšíření 3D Slicer **57**

B Nastavení vývojového prostředí pro rozšíření Supervised Segmentation Toolbox **58**

C Přidávání algoritmů pro předzpracování a klasifikaci **59**

D Obsah přiloženého CD **60**

SEZNAM OBRÁZKŮ

1.1 Ukázka ekvalizace histogramu. V levé části originální obraz a jeho histogram, v právé části obraz a jeho histogram po ekvalizaci. Histogramy byly přiblíženy z důvodu čitelnosti. 12
1.2 Ukázka gamma korekce. V levé části obraz zkreslený nelineární transformací a jeho histogram, v právé části obraz a jeho histogram po gamma korekci. Histogramy byly přiblíženy z důvodu čitelnosti. . . . 13
1.3 Ukázka mediánové filtrace. V levé části obraz zkreslený šumem typu pepř a sůl a v právé části obraz po mediánové filtraci. 14
1.4 Ukázka filtrace pomocí dolní propusti. V levé části obraz zkreslený gaussovským šumem a v právé části obraz po filtraci pomocí konvoluce s Gaussiánem. 14
1.5 Ukázka segmentace metodou rozvodí. Vlevo nepředzpracovaný obraz a výsledek jeho segmentace. V pravo obraz po předzpracování sobelovým operátorem a kvantizací na tři jasové hodnoty. Pod ním opět výsledek metody rozvodí. 15
2.1 Klasifikátor SVM. $\vec{u}$ je klasifikovaný vektor, $\vec{w}$ je vektor kolmý na hyperplochu, body P_i jsou pozitivní vzorky, N_i označují negativní vzorky. 18
2.2 Algoritmus náhodný les. 21
4.1 Schéma operace modulu. 26
4.2 Hierarchická návaznost a rozdělení tříd do funkčních skupin. 28
4.3 Uživatelské rozhraní pro načítání obrazů a nastavení předzpracování pro danou sadu obrazů. 29
4.4 Ukázka předzpracování pomocí mediánové filtrace. Uprostřed nastavovací grafické rozhraní. 30
4.5 Ukázka předzpracování Gaussova filtru. 30
4.6 Ukázka předzpracování pomocí Sobelova operátoru. 31
4.7 Diagram učení (vlevo) a klasifikace (vpravo). 32
4.8 Uživatelské rozhraní pro učení. 33
4.9 Uživatelské rozhraní pro klasifikace. 33
4.10 Ukázka výsledku vyhledávání učebních parametrů. 34
5.1 Řez 88 LG_0001_T1C (vlevo) a LG_0001_T2 35
5.2 C SVM - vliv vybraných typů obrazů na senzitivitu a specificitu . . . 36
5.3 C SVM - hledání parametrů, sensitivita 37
5.4 C SVM - hledání parametrů, specificita 37
5.5 C SVM - hledání parametrů, součet sensitivity a specificity 38

5.6 C SVM - vliv počtu řezů na sensitivitu, specificitu a počet učebních vektorů . . . 39
5.7 N SVM - vliv vybraných typů obrazů na senzitivitu a specificitu . . . 40
5.8 N SVM - hledání parametrů, sensitivita . . . 41
5.9 N SVM - hledání parametrů, specificita . . . 41
5.10 N SVM - hledání parametrů, součet sensitivity a specificity . . . 42
5.11 N SVM - vliv počtu řezů na sensitivitu, specificitu a počet učebních vektorů . . . 43
5.12 Náhodný les - vliv vybraných typů obrazů na senzitivitu a specificitu 44
5.13 Náhodný les - hledání parametrů, vlevo vliv velikosti uzlů a počtu stromů . . . 45
5.14 Náhodný les - hledání parametrů, vlevo vliv procenta vzorků mimo učební podmnožinu jednotlivých stromů (Oob) a počtu náhodných atributů v uzlech . . . 45
5.15 Náhodný les - vliv počtu řezů na sensitivitu, specificitu a počet učebních vektorů . . . 46
5.16 Srovnání výsledků N SVM a dvojitého prahování. Vlevo nahoře výsledek segmentace pomocí N SVM, vlevo dole pravdivostní obraz. Vpravo nahoře ručně dvojitě prahovaný obraz. Vpravo dole dvojitě prahovaný obraz s nejlepším číselným výsledkem součtu sensitivity a specificity. . . . 47

ÚVOD

Segmentace tomografických obrazů je vhodným způsobem automatického rozpoznávání struktur pro posouzení anatomických souvislostí i kvantifikaci velikostí například nádorových útvarů nebo funkčnosti orgánů. Klasické segmentační algoritmy ovšem trpí specifičností k jednomu typu úlohy a také zdlouhavým nastavováním pro dostatečnou přesnost. Tyto algoritmy jsou také citlivé na použité předzpracování. Z těchto a dalších důvodů jsou stále častěji prozkoumávány algoritmy strojového učení, které by tyto nedostatky odstraňovaly. Náhodné lesy (v originále Random Forests), Support Vector Machines (dále SVM), nebo umělé neuronové sítě jsou jedním z využitelných přístupů ke klasifikaci obrazů. Tyto algoritmy využívají automatickou statistickou, nebo matematickou analýzu klasifikovaných dat.

Platforma 3D Slicer obsahuje dobrý základ pro obecné zobrazení a zpracování až čtyřrozměrných medicínských obrazových dat. Aktivní komunita vývojářů a výzkumných pracovníků vytváří stále nová rozšíření a přidává tak do platformy novou funkcionalitu. Rychlý modul pro klasifikaci pomocí algoritmů strojového učení s dostatečným množstvím nastavení ovšem stále není dostupný.

Cílem diplomové práce je kromě literární rešerše v oblasti segmentace tomografických snímků a využití přístupů strojového učení k segmentačním úlohám také návrh a vývoj rozšíření platformy 3D Slicer pro segmentaci s využitím strojového učení. Literární rešerši se věnuje kapitola 1 a 2. Součástí druhé kapitoly je také seznam volně dostupných knihoven strojového učení, psaných v jazyce C++. Kapitola 3 představuje platformu 3D Slicer s otevřeným zdrojovým kódem, vysvětluje principy, na kterých je založena a popisuje rozšiřitelnost kódem třetí strany. V kapitole 4 jsou popsány požadované cílové vlastnosti rozšíření 3D Slicer a způsob, jakým byly dosaženy včetně programového návrhu. Popis testování a dosažené výsledky vytvořeným rozšířením lze nalézt v kapitole 5. Poslední kapitola 6 obsahuje zamyšlení nad využitelností vybraných algoritmů obecně a nad vhodností vyvíjeného rozšíření pro úkol segmentace tomografických dat.

V přílohách A, B a C nalezne případný zájemce krátké návody na kompilaci 3D Slicer, kompilaci vytvořeného rozšíření a přidávání funkcionality do rozšíření. Poslední dva návody jsou taktéž zveřejněny spolu se zdrojovým kódem rozšíření na repozitáři na serveru GitHub.

1 SEGMENTACE TOMOGRAFICKÝCH DAT

Segmentace je operace rozdělení obrazu na homogenní oblasti z hlediska vybraného parametru z důvodu snazší analýzy. Segmentovaný obraz pak může být analyzován buď manuálně, nebo automaticky. Parametrem segmentace může být například barva, intenzita, textura a jiné. Tomografickými daty se rozumí data snímaná po částech (řezech, sekcích) zobrazovaného objektu. Tento způsob získávání tomografických dat se využívá v širokém spektru průmyslových, lékařských i výzkumných oborů. Z lékařských tomografů lze zmínit výpočetní tomografii, magnetickou rezonanci, jednofotonovou emisní výpočetní tomografii nebo optickou koherentní tomografii.

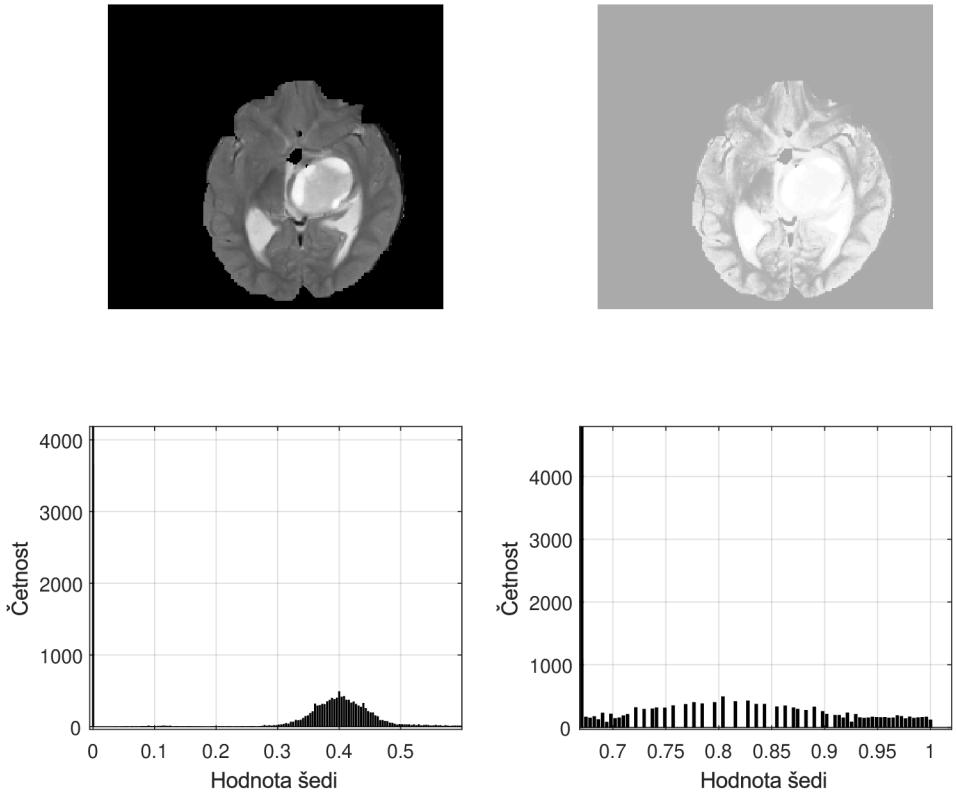
Před samotnou segmentací je mnohdy třeba obraz předzpracovat. Předzpracováním se snažíme negovat nehomogenity snímací apertury a připravit obraz pro další algoritmy.

1.1 Předzpracování

Tomografická data jsou často zarušena různými typy šumu. Obsahují artefakty, vzniklé při snímání i rekonstrukci obrazu. Jsou geometricky deformována a mají lineárně i nelineárně transformovaný kontrast. Tyto informace korelující s aktuálním okolím snímaného objektu a snímací aperturou jsou při segmentaci nežádoucí, jelikož jejich vliv je s ohledem na zobecnění pro jakýkoliv snímací přístroj a jakékoliv okolnosti snímání stochastický. Tyto prvky je třeba odstranit, čímž se zabývájí algoritmy předzpracování obrazu. Předzpracování obrazu nezvyšuje entropii, tedy nezvyšuje množství informací obsažených v obraze. Často naopak informaci ztrácíme. Tato ztráta informace může být žádoucí pro zrychlení výpočtu konkrétního algoritmu i pro zvýšení jeho přesnosti odstraněním zmíněných stochastických informací.

Transformace kontrastu jsou vhodné k získání optimálního obrazu se stejným zastoupením všech odstínů šedé. Příkladem takové transformace je například ekvalizace histogramu, kde maxima jsou roztažena do okolních hodnot, jak je možné vidět na Obr. 1.1. Vylepšením ekvalizace histogramu může být adaptivní transformace kontrastu, kde je histogram ještě před samotnou ekvalizací rozdělen na subhistogramy [3][4].

Gama korekce je způsob kompenzace nelinearity zobrazovacích prvků. Jedná se o inverzní funkci k nelineární transformaci kontrastu (obr. 1.2). Adaptivní gama korekcí[5] spolu s homomorfní filtrací byly získány kvalitně předzpracované obrazy týmem Tiwari et al[6]. Jiné transformace mohou například vytvářet negativní obraz,

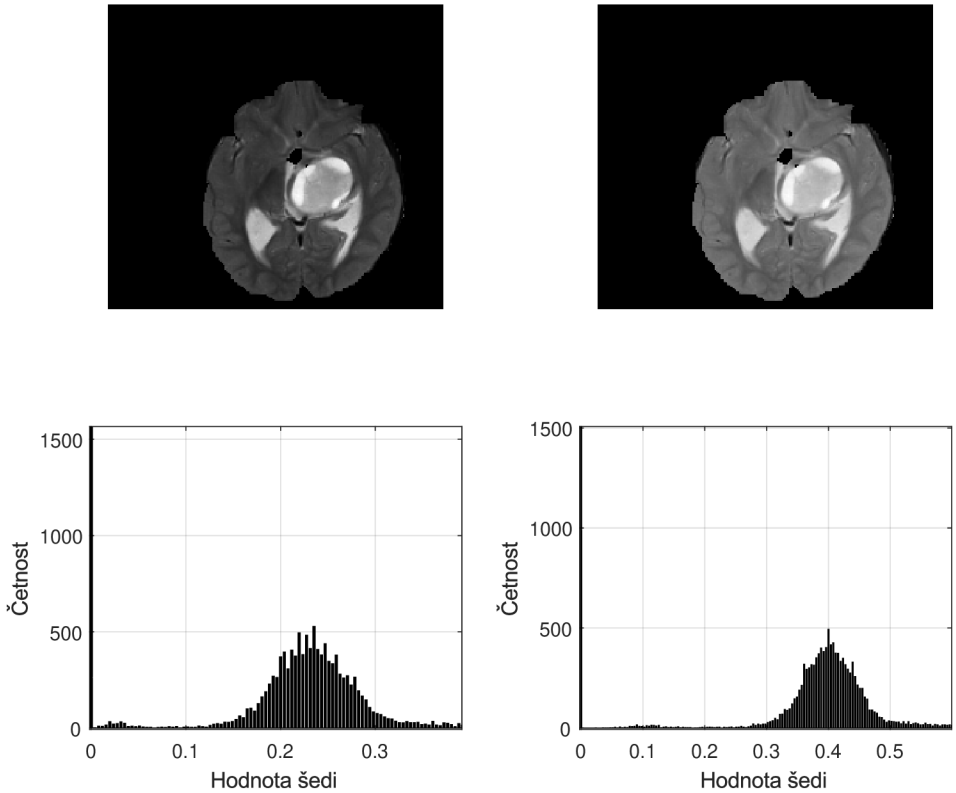


Obr. 1.1: Ukázka ekvalizace histogramu. V levé části originální obraz a jeho histogram, v právé části obraz a jeho histogram po ekvalizaci. Histogramy byly přiblíženy z důvodu čitelnosti.

pseudobarevný obraz, nebo binární obraz dle použité transformační funkce.

K odstranění gaussovského šumu, nebo šumu typu pepř a sůl je možné použít maskových operací, kde je nová hodnota v obrazového elementu vypočítána dle jeho okolí. Pro gaussovský aditivní šum je vhodné použít průměrovací masku, jejíž velikost je přímo úměrná rozmazání obrazu. Tento způsob předzpracování je pro ukázku na obr. 1.4. Rozmazání je možné negovat váženou průměrovací maskou, kde největší vliv na konečnou hodnotu bude mít centrální prvek. K odstranění šumu typu pepř a sůl lze použít místo průměru medián, jak je ukázáno na obr. 1.3. Výhodou takové nelineární filtrace je zdárné zachování hran v obraze. Masky lze taktéž použít jako konvoluční operátory a tím realizovat dále popsané filtrace.

Filtraci obrazu je možné uskutečnit v obrazovém i frekvenčním prostoru. V obrazovém prostoru se filtrace počítá pomocí 2D konvoluce s maskou. Ve speciálních případech je výpočet konvoluce obrazu s maskou separabilní operace, tedy se dá po-

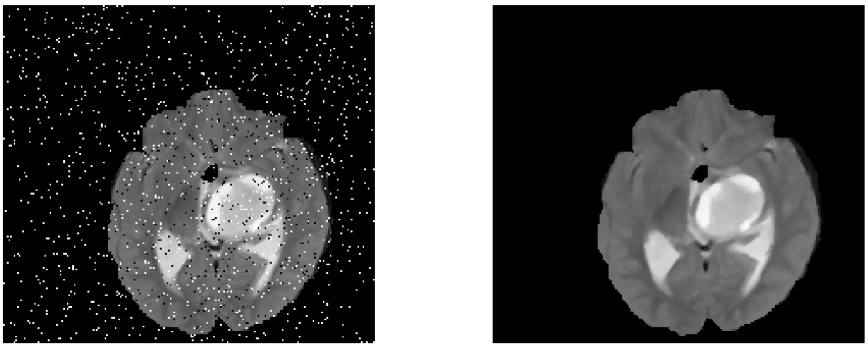


Obr. 1.2: Ukázka gamma korekce. V levé části obraz zkreslený nelineární transformací a jeho histogram, v právé části obraz a jeho histogram po gamma korekci. Histogramy byly přiblíženy z důvodu čitelnosti.

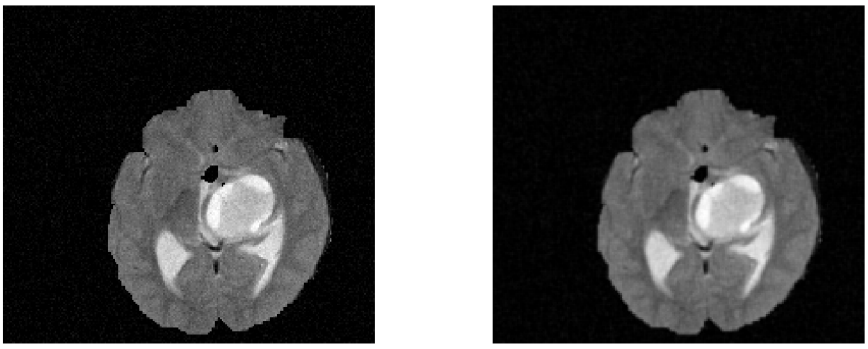
čítat v prvním průchodu pouze po řádcích, v druhém pouze po sloupcích. Místo $N \cdot N$ operací poté proběhne pouze $2N$, což snižuje výpočetní náročnost. Ve frekvenčním prostoru se filtrace provádí násobením spektra obrazu s frekvenční charakteristikou masky.

Těmito způsoby lze odstranit frekvenčně závislý šum, vytvořit první (Sobelův, Robertsův, Prewittové operátor), či druhou derivaci obrazu (Laplaciánem, Sobelovým Laplaciánem Gaussiánu), zaostřit (odečtením Laplaciánu), případně rozostřit (Gaussiánem) obraz. Operátory pro odhad první derivace lze získat hrubou hranovou reprezentaci, jelikož funkce jasu má v místech hran nejrychlejší změny.

Předzpracováním geometrickými transformacemi se snažíme odstranit geometrická zkreslení. Lze je dělit na rigidní a flexibilní. Rigidní transformace nemění vzdálenosti mezi jednotlivými prvky obrazu a řadí se mezi ně rotace a posunutí. Mezi flexibilní transformace patří změna měřítka a zkosení. Všechny zmíněné geometrické



Obr. 1.3: Ukázka mediánové filtrace. V levé části obraz zkreslený šumem typu pepř a sůl a v právé části obraz po mediánové filtraci.



Obr. 1.4: Ukázka filtrace pomocí dolní propusti. V levé části obraz zkreslený gaussovským šumem a v právé části obraz po filtraci pomocí konvoluce s Gaussiánem.

transformace tvoří transformaci afinní, tedy obecnou lineární transformaci.

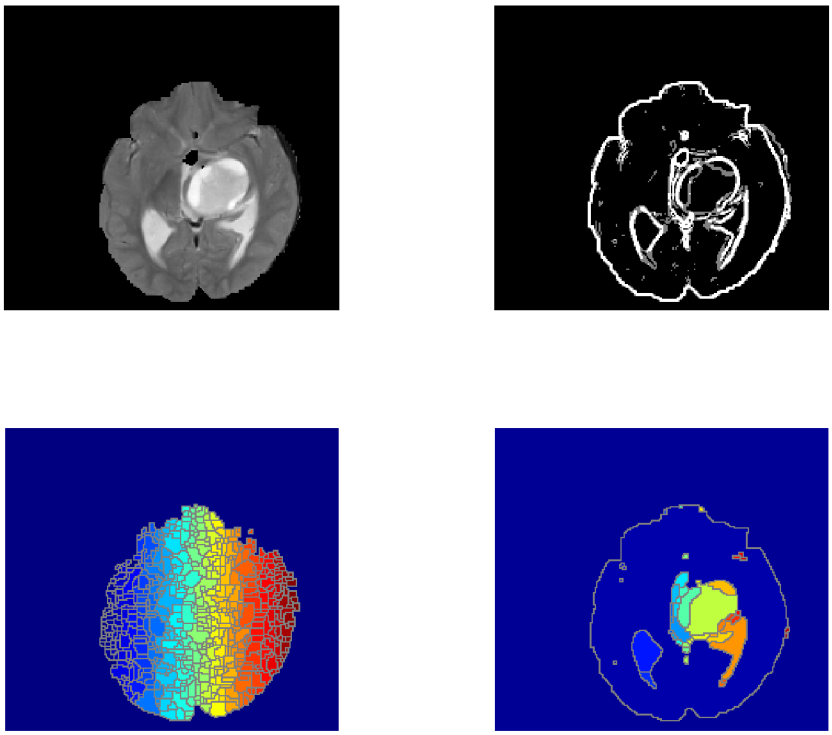
1.2 Segmentační algoritmy

Prahování je jedním z nejjednodušších druhů segmentačních algoritmů. Obraz je rozdělen na oblasti dle vybraných prahů jasu. V nejjednodušším případě je vybrán jeden práh a všechny obrazové elementy (pixely, voxely) s hodnotou jasu pod prahem jsou součástí jednoho segmentu a všechny nad prahem jsou součástí druhého elementu. Výběr prahů je možné také automatizovat například hledáním lokálních minim v histogramu. Často je segmentace prahováním využita ve spolupráci s jiným

segmentačním algoritmem. V [8] je prahování využito k vytvoření binárního obrazu pro extrakci kontury CT snímku.

Dalším segmentačním algoritmem je dělení a slučování oblastí. V první fázi tohoto algoritmu se obraz rozdělí například na kvadranty, u kterých se zkoumá homogenita dle vybraného kritéria. Oblasti, které nevyhovují kritériu se dělí dále, dokud všechny oblasti nejsou homogenní. V druhé fázi se slučují sousední oblasti, pokud vyhovují kritériu homogenity.

Nárůstání oblastí začíná ve vybraných bodech, jejichž okolí se testuje při každé iteraci algoritmu podmínkou homogenity. Výpočet řešení probíhá, dokud existují změny mezi obrazy generovanými po sobě jdoucími iteracemi.



Obr. 1.5: Ukázka segmentace metodou rozvodí. Vlevo nepředzpracovaný obraz a výsledek jeho segmentace. V pravo obraz po předzpracování sobelovým operátorem a kvantizací na tři jasové hodnoty. Pod ním opět výsledek metody rozvodí.

Obraz si lze představit jako vodu propustný (například perforovaný) reliéf, kde výška v každém bodě odpovídá hodnotě sledovaného parametru (typicky jasu). Pokud bude tento reliéf pomalu ponořován, tak v první fázi se zaplní segmenty, které na

sebe nenaléhají. Při dalším ponořování je třeba si zapamatovat místa setkání dvou takových oblastí. To jsou hrany výsledných segmentů, vytvořené metodou rozvodí. Algoritmus vyžaduje dostatečné předzpracování obrazu, jak ilustruje obr. 1.5.

Houghova transformace se využívá k vyhledávání matematicky popsaných objektů v obraze. Vstupem bývá hrubá hranová reprezentace. Každý bod může být považován za součást hledané křivky. Houghovou transformací se vytváří obraz, kde se pro každý vybraný bod vstupního obrazu sestrojí všechny možné hledané křivky, tímto bodem procházející. Pokud by byla hledanou křivkou přímka, tak se pro jeden bod vytvoří Houghovou transformací hvězda, případně kruh o nekonečném poloměru, dle míry přesnosti. Když se k tomuto bodu nalezne bod na jedné přímce v originálním obraze, tak v transformovaném obraze úměrně rozsvítí právě tato přímka.

Je-li známa hrubá pozice hranic segmentu, lze použít aktivních kontur. Pro tento výpočet je prvně nutné v každém bodě obrazu vypočítat jeho energii, dle požadovaných vlastností výsledného obrazu. Kontura pak migruje po energetických gradientech tak dlouho, dokud součet energie v bodech, které vyplňuje, neklesne na minimum. Kombinací aktivních kontur a narůstání oblastí je možné například segmentovat levou a pravou srdeční komoru z CT snímku srdce[7].

2 STROJOVÉ UČENÍ

Algoritmy strojového učení mají schopnost nalézat dříve neznámé spojitosti v učebních datech a využívat tuto znalost k odhadům vlastností neznámých dat. Jedno z jejich možných použití je řešení segmentačních problémů. Existují algoritmy pro použití s učitelem i bez něj, ovšem při segmentaci se téměř výhradně používá učení s učitelem. K tomu se využívá matematické optimalizace, počítané numericky, vyjímečně v jednoduchých případech analyticky. Následuje výběr přístupů ke strojovému učení, které jsou vhodné a často používané ke klasifikaci medicinských obrazů.

2.1 Umělé neuronové sítě

Neuronové sítě jsou inspirované zjednodušeným modelem biologických nervových systémů, například lidského mozku. Neuron, jako základní stavební jednotka umělé neuronové sítě má vstupy modifikované vahami a výstup, který může být binární v případě skokových funkcí, nebo spojitý v případě lineární kombinace vstupů a v případě sigmoidálních funkcí. Matematicky lze neuron zapsat jako

$$y = \varphi(\sum_{i=0}^{m} w_i \cdot x_i), \tag{2.1}$$

kde y je výstup neuronu, φ je přenosová funkce neuronu (zmíněná skoková, sigmoidální a jiné), i je počet vstupů, w jsou váhy na jednotlivých vstupech a x jsou vstupy. Lze odvodit, že při použití skokové přenosové funkce dělí neuron klasifikační prostor hyperpolochou orientovanou dle vah vstupů. Pro lineárně neseparovatelné problémy je potřeba více neuronů.

Neurony se organizují do vrstev, kde výstupy z jedné vrstvy figurují jako vstupy vyšší vrstvy. Větší počet vrstev výrazně zhoršuje interpretovatelnost neuronové sítě a zvyšuje výpočetní náročnost. Velkým množstvím tzv. skrytých vrstev se vyznačuje oblast strojového učení zvaná hluboké učení (Deep Learning) a používá se k modelování abstraktních spojitostí vstupních dat.

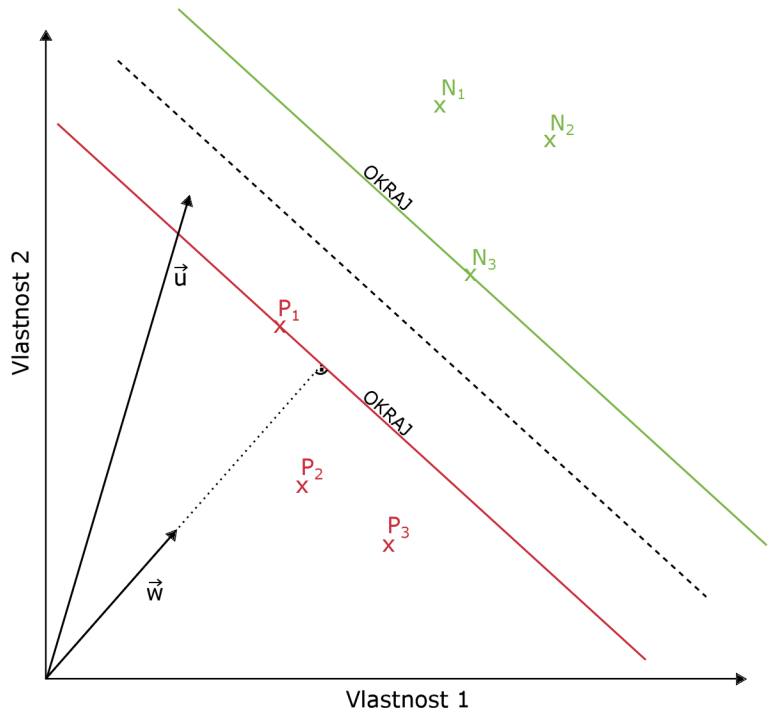
Algoritmus zpětného šíření chyby trénuje umělou neuronovou síť při učení s učitelem. Nejprve jsou do sítě vložena data běžným způsobem a při získání výsledku je ten porovnán se známým správným výsledkem. Jejich rozdíl je zpětně propagován od nejvyšších vrstev a váhy jednotlivých neuronů jsou upravovány směrem ke kýženému výsledku.

Jednou z největších nevýhod neuronových sítí je konvergence řešení k lokálním minimům zvláště u komplexních případů. Za další nevýhodu by se dala považovat nemožnost paralelizace několika vrstev, jelikož každá vrstva musí čekat na dokončení

výpočtu celé předchozí. Ovšem v jednotlivých vrstvách lze s úspěchem paralelizovat jednotlivé neurony, což z umělých neuronových sítí dělá ideální kandidáty pro výpočty na grafických kartách.

2.2 Support Vector Machines

SVM je klasifikátor s první zmínkou v literatuře až v roce 1976[9] a v prvních používaných verzích byl limitován pouze na dvě lineárně separovatelné třídy. Této klasifikace dosahuje oddělením skupin objektů hyperplochou, která má od nejbližších objektů obou tříd co největší vzdálenost. Hyperplochy rovnoběžné s oddělovací, které zároveň protínají nejbližší objekty se jmenují okraje. Support vector, neboli pomocný vektor je vektor, mířící k bodům každé třídy, které leží na okrajích, tedy vektory do N_3 a P_1 na obr. 2.1.



Obr. 2.1: Klasifikátor SVM. $\vec{u}$ je klasifikovaný vektor, $\vec{w}$ je vektor kolmý na hyperplochu, body P_i jsou pozitivní vzorky, N_i označují negativní vzorky.

Aby bylo možné zjistit, do které ze tříd daný objekt patří, je třeba ho vyjádřit jako N-rozměrný vektor $\vec{u}$ v souřadném systému definovaném vybranými parametry objektu. Tento vektor je promítnut skalárním součinem s $\vec{w}$, což je vektor kolmý na diskriminační hyperplochu. Rozhodovací pravidlo poté získáme jako

$$\vec{u} \cdot \vec{w} + b \geq 0, \tag{2.2}$$

kde b je vzdálenost hyperplochy od počátku souřadnic. V případě splnění podmínky je objekt definovaný $\vec{u}$ považován za součást jedné třídy, jinak náleží druhé třídě. V pravidle prozatím není vyznačen projev okrajů. Rozteč okrajů definujeme jako ± 1 a úpravou rovnice 2.2 získáme

$$\begin{aligned} \vec{x_1} \cdot \vec{w} + b &\geq 1, \\ \vec{x_2} \cdot \vec{w} + b &\leq -1, \end{aligned} \tag{2.3}$$

kde $\vec{x_1}$ a $\vec{x_2}$ jsou vektory odpovídající objektům v první a druhé třídě. Rovnice lze zjednodušit vynásobením y_i, které je rovno $+1$ pro vzorky z první třídy a -1 pro vzorky z druhé třídy.

$$y_i \cdot (\vec{x_i} \cdot \vec{w} + b) - 1 \geq 0. \tag{2.4}$$

Rovna nule bude tato nerovnice v případě, že vybraný bod leží na okrajích. K výpočtu šířky okrajů D lze využít známých $\vec{x_1}$ a $\vec{x_2}$, ležících na okrajích, jejichž rozdíl se rotuje jednotkovým vektorem kolmým k okrajům.

$$D = (\vec{x_1} \cdot \vec{x_2}) \cdot \frac{\vec{w}}{|\vec{w}|}. \tag{2.5}$$

Dosazením z 2.4 se rovnice zjednoduší na

$$D = \frac{2}{|\vec{w}|} \tag{2.6}$$

a tím je patrné, že minimalizace $|\vec{w}|$ vede k maximalizaci šířky okrajů a tedy k optimálnímu řešení. Vložená nerovnice 2.4 ovšem znemožňuje hledání extrémů prvními derivacemi. Vynásobením s Lagrangeovými multiplikátory se toto omezení odstraní. Tím a drobnou úpravou rovnice 2.6, která zjednoduší pozdější rovnice získáme

$$L = \frac{1}{2}|\vec{w}|^2 - \sum \alpha_i \cdot [y_i \cdot (\vec{w} \cdot \vec{x_i} + b) - 1], \tag{2.7}$$

kde α_i označuje Lagrangeovy multiplikátory. Pro hledání extrémů získáme

$$\begin{aligned} \frac{\partial L}{\partial \vec{w}} &= \vec{w} - \sum \alpha_i \cdot y_i \cdot \vec{x_i} = 0, \\ \frac{\partial L}{\partial b} &= \sum \alpha_i \cdot y_i = 0. \end{aligned} \tag{2.8}$$

Dosazením horních výrazů do 2.7 a zjednodušením

$$L = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j}. \tag{2.9}$$

Tento kvadratický optimalizační problém již závisí pouze na skalárním součinu párů vstupních vzorků, jelikož y i α je pro daná $\vec{x}$ konstantní. Podobně, dosazením výrazu pro $\vec{w}$ z 2.8 do 2.2 závisí rozhodovací pravidlo také na skalárním součinu

neznámého vektoru $\vec{u}$ a známého $\vec{x_i}$. Pro malé množství dat lze optimalizaci vyřešit analyticky, ovšem v praxi se téměř vždy řeší numericky.

Matematické vyjádření SVM v tomto smyslu ovšem stále trpí nemožností oddělovat lineárně neoddělitelné problémy. Řešení takového omezení přišlo v podobě transformací jádrovými funkcemi, které ve většině případů problém přemění na problém ve vyšší dimenzi, kde již je lineárně separovatelný. Jednou z často užívaných jádrových funkcí je metoda vycházející z radiální bázové funkce. Tato funkce nelineárně transformuje vstupní data do prostoru o nekonečné dimenzi dle rovnice

$$h(x) = \sum_{n=1}^{N} w_n \cdot exp(-\gamma \cdot ||x - x_n||^2), \tag{2.10}$$

kde je aktuální bod x transformován jako součet příspěvků každého jiného bodu. Parametr γ udává míru příspěvků jednotlivých vzdáleností a váhy w_n jsou závislé na funkčních hodnotách v bodech x.

Druhým problémem je, že SVM je binární klasifikátor a jako takový odděluje pouze dvě třídy. Toto lze vyřešit použitím více SVM, kde každý odděluje jednu třídu od zbytku.

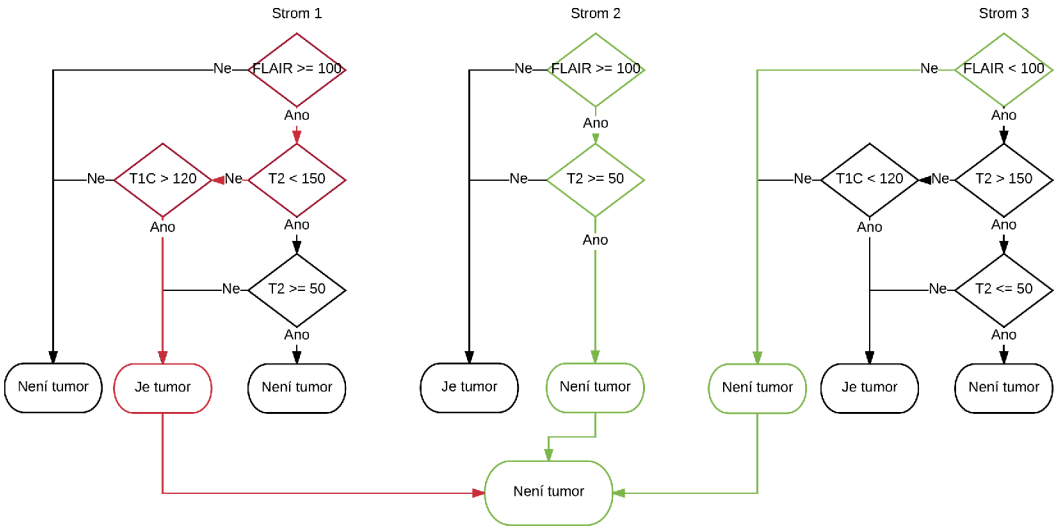
Výhodou SVM je, že se jedná a konvexní optimalizační problém a tedy vždy konverguje do globálního minima, tedy optimálního řešení.

Reálné použití algoritmu navíc ztěžuje výskyt šumových a výjimečných hodnot, které zapřičiňují zúžení okrajů a případně znemožňují nalézt jakékoliv řešení. Toto je vyřešeno tzv. Soft-Margin SVM, tedy SVM s měkkými okraji. Tento typ umožňuje některým extrémním učebním vzorkům nebýt pokládany za pomocné vektory. Dvě nejpoužívanější implementace tohoto algoritmu jsou C-SVM[24] s parametrem C, kterým se nastavuje rozvolněnost okraje (čím vyšší C, tím se klasifikátor blíží klasickému SVM) a N-SVM[25] s parametrem N, který udává horní limit maximálně špatně klasifikovaných vzorků a zároveň dolní limit počtu pomocných vektorů ve zlomku všech učebních vzorků. Zatímco N je výrazně jednodušší na interpretaci, C se snáze optimalizuje.

2.3 Náhodný les

Cílem algoritmu náhodného lesa (obr. 2.2) je vytvoření skupiny nekorelovaných rozhodovacích stromů k odstranění závislostí na náhodnných chybách učebního souboru[22]. Rozhodovací strom je soubor řetězených pravidel když-pak, na jehož konci je vstup zařazen do některé ze tříd. Jedná se o přístup ke strojovému učení s učitelem. K učení je tedy k dispozici soubor známých výsledků. Dekorelace rozhodovacích stromů se dosahuje náhodným výběrem části učebního souboru, ze kterého se strom vytvoří. Lze náhodně vybírat i vlastnosti jednoho učebního prvku. Výsledek klasifikace je

počet stromů, které hlasovaly pro jednotlivé třídy a k jednoznačné klasifikaci lze vybrat například třídu s největším počtem hlasů.



Obr. 2.2: Algoritmus náhodný les.

2.4 Learning Classifier System

Tento nešťastně pojmenovaný přístup ke strojovému učení kombinuje inferenční algoritmy umělé inteligence spolu s evolučními algoritmy k vytvoření komplexního klasifikátoru[23]. Architektura Learning classifier system je dělena na několik nezávislých celků. Populaci klasifikátoru tvoří opět pravidla když-pak a v první iteraci učení je populace prázdná. V dalším kroku probíha kontrola shody hodnot pravidel v populaci s učebním souborem. Příkladem budiž učební vektor, který tvrdí, že při teplotě 10 °C a nízkém tlaku bude pršet. Dále postoupí pouze pravidla posuzující buď teplotu 10 °C nebo nízký atmosférický tlak. Tato pravidla jsou dále rozdělena podle pravdivosti. Tedy pokud pravidlo tvrdí, že bude pršet, je přesunuto k pravdivým.

Pokud dále nepostoupí žádné pravidlo (nebo nízký počet), vytvoří se nová pravidla analýzou (v originále covering) učebního vektoru. Nepravdivá pravidla se opraví dle učebního vektoru. Je posouzena zdatnost jedinců v populaci stejně jako v klasických evolučních algoritmech a hodnoty jsou uloženy. Dále probíhá proces subsumpce, ve kterém se slučují pravidla tak, aby nepostoupila ta, u kterých existuje více přesné a více obecné pravidlo. Křížením, mutacemi a dalšími genetickými algoritmy se získá nová generace pravidel a proces se opakuje. Na konci učení lze eliminovat málo přesná a příliš striktní pravidla pro snazší interpretaci a rychlejší klasifikaci.

Algoritmus je relativně nový v porovnání s předchozími (první zmínky na počátku osmdesátých let) a stále nejsou k dispozici plně optimalizované, rychlé a rozšiřitelné implementace.

2.5 Volně dostupné knihovny

Knihoven algoritmů strojového učení, které lze při vývoji aplikací použít, je značné množství[11]. Následující výpis obsahuje výběr těch nejužívanějších, psaných v jazyce C++. Všechny vypsané knihovny nejsou závislé na platformě (nevyužívají platformně specifické funkce a pokud ano, tak obsahují všechny typy a výběr správné je při kompilaci dle stroje, na kterém se kompiluje).

Knihovna Dlib[12] má z dále popsaných největší záběr algoritmů, nejen strojového učení. Vznikla v době, kdy seznam C++ knihoven strojového učení nebyl zdaleka tak vyčerpávající. Je stále v aktivním vývoji a její záběr se rozšiřuje například o správu vláken, či XML analyzátor. V současnosti obsahuje i rozhraní v Pythonu. Knihovna je také poměrně dobře zdokumentovaná. Dlib obsahuje také diagram pro správný výběr algoritmu strojového učení.

Mlpack[13] je nejmladší knihovna z citovaných. Zajímavé je, že i když vznikla v době standardu C++11, který řešil do té doby neřešený problém řízení vláken, používá stále knihovnu Boost, která tento problém řešila do té doby. Přitom tvůrci standardu C++11 pouze vzali knihovnu Boost a její implementaci téměř beze změn přesunuli do std namespace[10]. Samotný mlpack na svých webových stránkách předvádí porovnání svých algoritmů s jinými knihovnami, kde konkurenci poráží většinou o několik řádů. Tvůrci také tvrdí, že knihovna byla vytvořena tak, aby byly její algoritmy vysoce škálovatelné. Nevýhodou knihovny je absence implementace algoritmu pro SVM. Knihovna je stále aktivně vyvíjena.

Shark[14], podobně jako předchozí, obsahuje implementaci metod strojového učení s učitelem, i bez něj. Metody založené na evolučních algoritmech, gradientních optimalizacích, či neuronové sítě. Jako jediní na svých webových stránkách myslí i na vývojářskou menšinu, tedy vývojáře na operačním systému Windows. Pro použití i zkompilování Sharku je jako v předchozích případech potřeba knihovna Boost. Shark je stále doplňován a vyvíjen během psaní této práce.

Caffe[15] je asi nejznámější knihovna strojového učení, používaná i Googlem v projektu Deep Dream[16]. Oproti ostatním zmíněným se zaměřuje především na Deep Learning neuronových sítí. Caffe lze použít i pro paralelizaci na grafických kartách od společnosti nVidia. Obsahuje i rozhraní pro použití v Matlabu, či Pythonu.

3 PLATFORMA 3D SLICER

3D Slicer je software pro zpracování a vizualizaci třírozměrných i čtyřrozměrných medicínských dat s otevřeným zdrojovým kódem, dostupný zdarma bez omezení. Je nezávislý na operačním systému díky použití neproprietárních knihoven. Naprogramovaný je jazykem C++ a využívá událostí pro komunikaci mezi rozhraními. Jeho nevýhodou je nemožnost použití v klinické praxi.[17]

Architekturu 3D Slicer je možné rozdělit do tří vrstev, pojmenovaných jádro, moduly a rozšíření (extensions).

Jádro je navrženo kolem známého a ověřeného vzoru Model-View-Controller, volně přeloženo jako model, zobrazení a ovládání. Uživatel vidí zobrazení a skrz ovladač ovládá data, obsažená v modelu. Změna dat vyvolá změnu zobrazení. Příkladem lze uvést segmentaci obrazových dat. Ovládáním se načte obraz do struktur v modelu a zobrazí se. Většinou v grafickém uživatelském rozhraní se opět pomocí ovládání nastaví vybraná metoda a spustí se procedura segmentace. Výsledky se zapíší do modelu a téměř okamžitě poté se změny projeví také ve zobrazení.

Moduly jsou interní součástí distribuce 3D Slicer a rozšiřují funkcionalitu jádra a mohou být závislé samy na sobě. Rozšíření oproti tomu jsou uživatelem instalovatelná. Rozšíření lze ovšem také chápat jako způsob zabalení modulů, které nemohou být zařazeny do distribuce, například z důvodu uzavřené licence.

3.1 Knihovny užívané platformou

Model jádra 3D Slicer je tvořen především pomocí knihovny Medical Reality Modeling Language (dále MRML). Ta reprezentuje hierarchickou strukturu medicínských dat a vazby mezi nimi a jejím základem je XML (Extensible Markup Language). Zobrazování je vyřešeno modulárními zobrazovači pomocí Qt Frameworku (dále Qt), konkrétně verze 4.8.6 a na hardwarové úrovni pomocí OpenGL. Ovládání (ve 3D Slicer označované jako Logic) probíhá manipulací s MRML strukturami, jejichž změny se promítnou do modelu.

Mezi další knihovny, používané 3D Slicer patří také Visualisation Toolkit (dále VTK), která obsahuje základní funkce pro počítačové vidění, 3D zobrazování a zpracování obrazů. Common Toolkit (dále CTK) obsahuje podporu The Digital Imaging and Communications in Medicine (dále DICOM) a specializované grafické prvky pro provázání 3D Slicer s tímto standardem. Insight Toolkit (dále ITK) je knihovna specificky navržená pro registraci a segmentaci obrazových dat.

Jak je možné vysledovat, některé knihovny se svými schopnostmi překrývají. Například Qt, VTK i MRML obsahují registrace a manipulace s událostmi. Tato redundance zaručuje použití správných prostředků pro konkrétní příležitosti.

3.2 Možnosti rozšíření

Rozšiřitelnost je jedna z hlavních vlastností 3D Slicer. Ať se jedná o rychlé prototypování ve skriptovacím jazyce Python, oddělené pomocné programy s rozhraním přes příkazovou řádku, či načitatelné s přímým přístupem k rozhraním 3D Slicer i pomocných knihoven. Vzor všech typů lze vytvořit pomocí modulu Extension Wizard.

3.2.1 Skriptovaná rozšíření

Nejsnadněji vytvořitelná jsou skriptovaná rozšíření v jazyce Python. Samotnou komunikaci mezi takovým rozšířením a jádrem 3D Slicer zajišťují zabalovací funkce. Nevýhodou tohoto způsobu rozšíření je nízká kontrola nad hardwarem a nepoměrně vysoké nároky na paměť i výpočetní čas[18][19], lze samozřejmě optimalizovat i kód v Pythonu[20], ovšem stále existují limitace zabalovacích funkcí. Naopak je to vhodný způsob k rychlému vytvoření a otestování prototypů.

3.2.2 Rozšíření s rozhraním přes příkazovou řádku

Rozhraní přes příkazovou řádku programátorovi umožňuje použít téměř jakýkoliv programovací jazyk při tvorbě svého rozšíření. Přímo ve 3D Slicer je takové rozšíření reprezentováno automaticky vygenerovaným grafickým uživatelským rozhraním. Nevýhodami tohoto způsobu jsou nemožnost přímo využívat jádro 3D Slicer a samozřejmě omezený počet znaků při volání rozšíření přes příkazovou řádku (maximum ve Windows 8191 znaků, na GNU/Linuxu se počet liší dle zvolené distribuce).

3.2.3 Načitatelná rozšíření

Poslední nabízenou možností je vytvoření načitatelného rozšíření v jazyce C++, zkompilovaného do formy dynamicky linkované knihovny. Tento způsob umožňuje případnému vývojáři největší kontrolu nad využitím systémových prostředků a umožňuje přímo využívat jádro 3D Slicer. Nevýhodou je nepoměrně vysoký nárok na vstupní znalosti[17], jelikož je potřeba znát vnitřní uspořádání kódu a jednotlivá rozhraní výše popsaných knihoven.

4 VÝVOJ A POUŽITÍ ROZŠÍŘENÍ

Z důvodu největší kontroly nad systémovými prostředky i nad během 3D Sliceru při používání modulu, a také možnosti využití některých součástí jádra 3D Slicer byla vybrána možnost rozšíření načitatelným modulem. Vývoj i testování bude probíhat na operačním systému Windows 10, ve vývojovém prostředí Visual Studio 2015 s kompilátorem msvc2015, který podporuje standard C++11.

Tato kapitola představí vytyčené cíle, které by měl modul segmentace tomografických obrazů splňovat. Dále představí anotace jednotlivých použitých tříd a funkcí s jejich hierarchickým provázáním. Poslední součástí je uživatelská příručka. V případě potřeby důkladnějšího popisu kódu v případě rozšiřování seznamu předzpracovacích algoritmů nebo klasifikátorů čtenář nalezne v příloze C podrobný návod.

4.1 Stanovené cíle

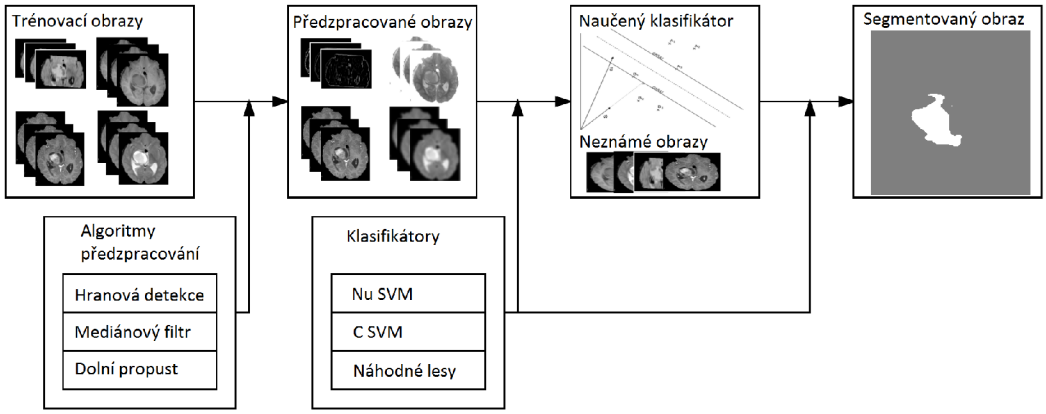
Modul by měl uživateli umožňovat výběr z několika používaných metod předzpracování obrazu. Minimálně by měl obsahovat mediánový a Gaussův filtr. Rozšíření by nijak nemělo omezovat možnosti načítaní typů obrazů, které již podporuje 3D Slicer.

Jako další úkol lze uvést výběr a nastavení klasifikačního algoritmu. Algoritmy budou vybrány průnikem množiny nejpoužívanějších v dostupných a citovaných literárních zdrojích a množiny algoritmů implementovaných ve vybraných knihovnách. Vybranou metodou budou předzpracovány obrazy označené jako rozsah souborů jednoho parametru, například všechny T1 váhované MRI obrazy. Bude možné přidávat neomezené množství těchto parametrických dat.

Všechny algoritmy, z kterých bude uživatel dále vybírat, jsou algoritmy strojového učení s učitelem a tedy kromě holých trénovacích dat potřebují také výsledek klasifikace. Počet souborů v každém řádku výběru parametrických obrazů z obrázku bude vyžadován stejný. Modul bude předpokládat, že všechna data jsou řazena stejně, tedy první parametrický obraz z prvního rozsahu odpovídá prvnímu parametrickému obrazu z druhého rozsahu a prvnímu pravdivostnímu obrazu.

Výsledek klasifikace ve formě natrénovaných koeficientů a pravidel bude možné uložit spolu s metadaty o použitém algoritmu, počtu a předzpracování vstupních obrazů.

Klasifikační část modulu opět nabídne neomezený výběr vstupních parametrů. V dalším kroku bude možné načíst klasifikátor definovaný uloženým souborem. Bude předvyplněn počet a předzpracování parametrických obrazů. Zjednodušené schéma použití modulu lze shlédnout na obr. 4.1.



Obr. 4.1: Schéma operace modulu.

Výpočetní časy jednotlivých metod by neměly být o mnoho delší, než při jejich volání pouze z knihovny strojového učení, bez přidaných nároků platformy 3D Slicer.

Grafické uživatelské rozhraní by mělo být přehledné, estetické a modul by se měl prezentovat srozumitelně jak pro vědecké pracovníky v oblasti strojového učení a segmentace obrazů, tak pro výzkumníky v oblastech medicíny.

Zdrojové kódy budou zveřejněny na webovém Git repozitáři hostovaném serverem GitHub a pokud splní podmínky dané 3D Slicer komunitou, bude jej možné stáhnout i z GUI 3D Slicer. Samotný kód by měl být čistý, čitelný, dělený do přirozených a očekávaných celků a v komplexních částech řádně zdokumentovaný.

4.2 Základní struktura kódu

Na úrovni komunikace s prostředím 3D Slicer lze rozdělit modul na dva logické celky. Třída `SegmentationToolboxTrainingWidget` obsahuje GUI pro trénovací část modulu včetně GUI třídy `ImageRangePreprocessingSelector`, která umožňuje sekvenční výběr neomezeného počtu sad obrazů a jejich předzpracování. Obsahuje seznam předzpracovacích algoritmů, seznam klasifikátorů a vlastní kopii třídy `VolumeManager`, která má za úkol správu a provázání souborů s třídami a funkcemi, které vyžadují přístup k datům obrazů. Drží si také ukazatel na aktuálně vybraný klasifikátor z důvodu změny GUI při změně klasifikátoru. Tato třída obsahuje také konečnou agregaci serializací předzpracovacích algoritmů a klasifikátoru a nabízí možnost tuto agregaci uložit do souboru s příponou stc, dle anglického Supervised Toolbox Classifier. Soubor stc je strukturován do bloků. V prvním bloku je vždy popsán název klasifikátoru a jeho serializovaná verze. Ve dalších blocích je popsána každá sada obrazů (jedna dimenze učebních vektorů) výpisem souborů a použitým předzpracováním. Příklad souboru je možné shlédnout ve výpise 4.1. Výpis zobrazuje

jediný algoritmus, který provádí serializaci do textové podoby. Kromě náhodného lesu knihovny shark se všechny klasifikátory serializují do binární podoby.

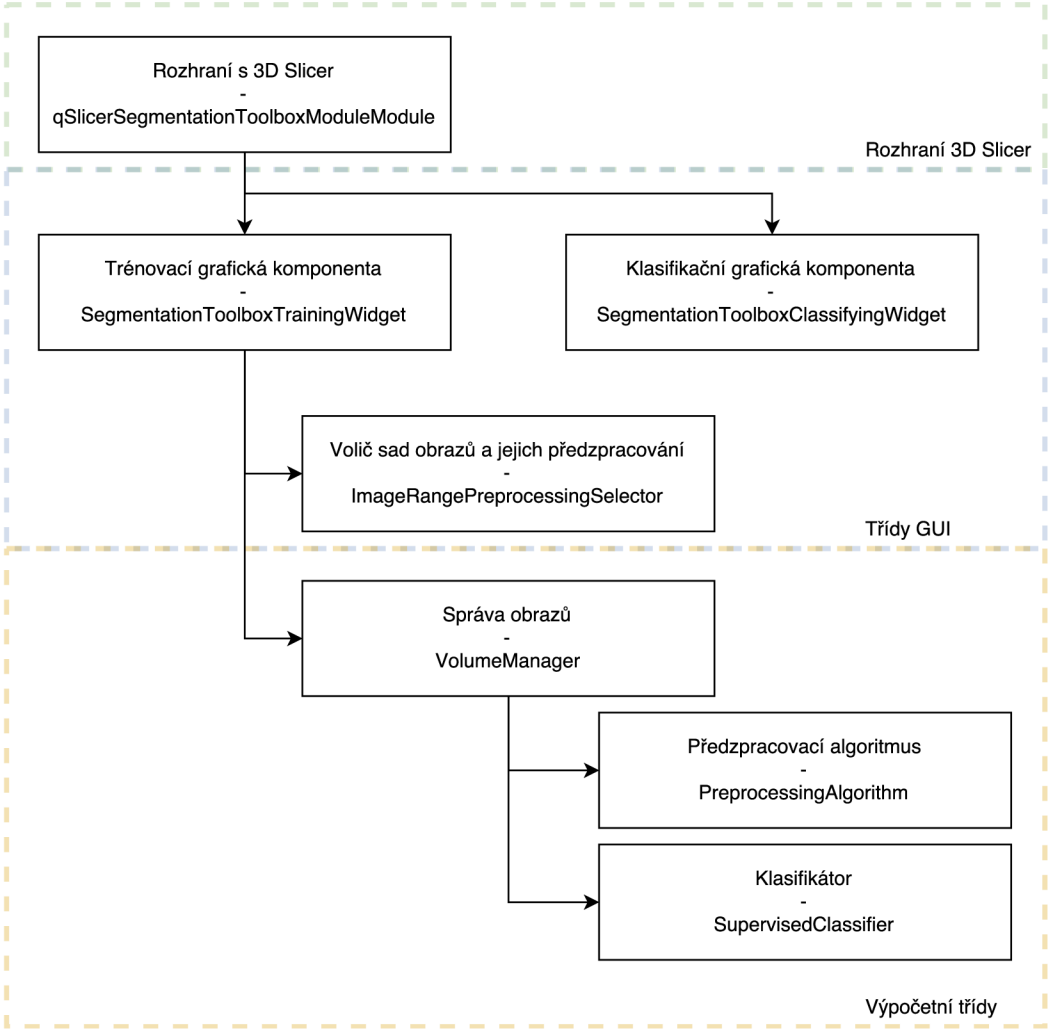
Výpis 4.1: Ukázka stc souboru

```
clasifierName=Random Forest - Shark
===classifierSetting===
22 serialization::archive 14 1 0
0 2 7.43514117210533620e-03 3.76611903266677430e-03
1 2 -2.27510437503552332e+00 -2.25389505504245680e+00 1 1 0
2 370 0 0 0 1 0
3 93 0 0 0 0 1 3.37194839432283899e-01 1 2
4 0 0 0  1 0  0 0
5 2 1 0  0 0 2 0 -8.25251846474982731e-01 3 4
6 0  0  3 0  0 0
7 2 1.0 0  0 0 4 1 1.35781309728497979e+00 5 92
8 0  0  5 0 2.53543196331662912e+00 6 91
9 0  0  6 0 -8.17816705302877534e-01 7 8
===/classifierSetting===
fileCount=2
fileList=D:\CestaKSouboru\BRATS_LG001_T1C.mha,D:\CestaKSouboru\BRATS_LG002_T1C.mha
preprocessingAlgorithm=3D Median Filter
===preprocessingSetting===
maskSizeX=2
maskSizeY=2
maskSizeZ=2
===/preprocessingSetting===
fileCount=2
fileList=D:\CestaKSouboru\BRATS_LG001_T2.mha,D:\CestaKSouboru\BRATS_LG002_T2.mha
preprocessingAlgorithm=None
===preprocessingSetting===
===/preprocessingSetting===
```

Druhým z logických celků je třída `SegmentationToolboxClassifyingWidget`, která je téměř ekvivalentní s `SegmentationToolboxTrainingWidget`. Místo agregace serializací ovšem obsahuje inverzní deserializaci a dle načteného souboru vytvoří klasifikátor a nastaví počet a předzpracování obrazových sad.

Tyto třídy jsou sdruženy v třídě vygenerované platformou 3D Slicer při vytváření šablony rozšíření s názvem `qSlicerSegmentationToolboxModuleModule`. Právě tu vyhledává část 3D Slicer, která má za úkol načítání a spouštění načitatelných rozšíření. Schéma rozdělení uvedených tříd do funkčních skupin a jejich hierarchická návaznost je zobrazena obr. 4.2. Z důvodu přehlednosti schéma neobsahuje rozepsanou hierarchii klasifikační grafické komponenty, neboť je totožná s hierarchií učební grafické komponenty.

Struktura repozitáře je téměř totožná s vygenerovanou složkovou hierarchií šablony. V kořenovém adresáři lze nalézt složku *lib* s kompilovaným 64-bitovým rozšířením pro operační systém Windows odpovídající verze 3D Slicer. Další složkou v kořenovém adresáři je *SegmentationToolboxModule*, kde je obsažen zdrojový kód, dělený do několika podsložek. Kompatibilita zdrojového kódu napříč operačními systémy je zaručena příslušnými *CMakeLists.txt* soubory pro použití softwarem CMake.

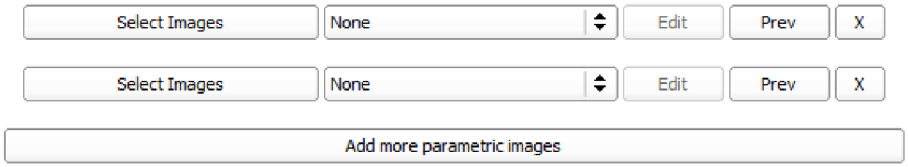


Obr. 4.2: Hierarchická návaznost a rozdělení tříd do funkčních skupin.

4.3 Práce se soubory

Načítání souborů je uživateli prezentováno skrz třídu `ImageRangePreprocessingSelector`, jejíž grafická reprezentace je zobrazena na obr. 4.3. Na obrázku je třída ve stavu, kdy lze vybrat dvě různé sady obrazů, například několik T1 váhovaných obrazů v první sadě a několik T2 váhovaných v sadě druhé. Další sady (zobrazené jako řádky) lze přidat kliknutím na tlačítko „Add more parametric images“.

Každý vytvořený řádek obsahuje tlačítko pro vyvolání dialogu pro výběr souborů. Dialog je závislý na operačním systému a v případě Windows je to Windows File Explorer. Dále lze volit z rolovacího seznamu algoritmus předzpracování a tlačítkem „Edit“ zobrazit grafické rozhraní pro jeho nastavení. Tlačítkem „Prev“ je možné zkontrolovat výsledek nastaveného předzpracování na prvním obrazu ze zvolené sady. Křížek umožňuje uživateli odstranit celou příslušnou sadu i s nastavením. Třída



Obr. 4.3: Uživatelské rozhraní pro načítání obrazů a nastavení předzpracování pro danou sadu obrazů.

nakonec obsahuje metodu pro odeslání cest k vybraným souborům a ukazatele na nastavené algoritmy předzpracování.

Jak již bylo řečeno v kapitole 3, obrazová data jsou ve 3D Slicer reprezentována MRML uzly, které jsou součástí scény. Správa scény, vytváření a mazání uzlů má za úkol třída `VolumeManager`. Obsahuje metody pro vícevláknové použití předzpracujících algoritmů a předzpracovaná data odesílá k učení klasifikátorů, případně ke klasifikaci. Také umožňuje zobrazit náhled předzpracovaného obrazu.

4.4 Předzpracovací algoritmy

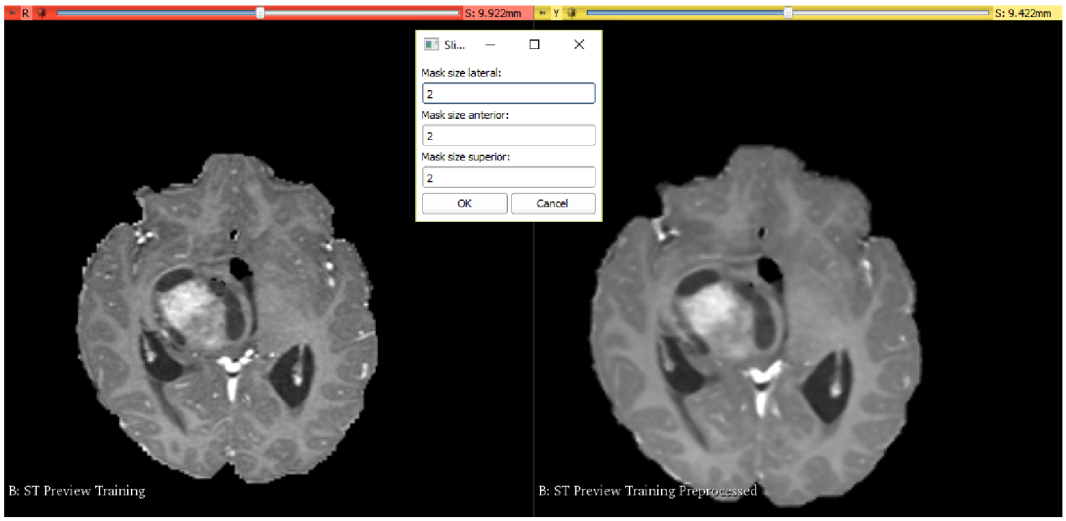
Všechny tyto algoritmy dědí z virtuální třídy `PreprocessingAlgorithm`. Ta vyžaduje přetížení metod pro serializaci a deserializaci a předzpracující funkce. Obsahuje možnost vytvořit dialogové okno pro nastavování a nastavit jméno.

Aktuálně jsou implementovány tři algoritmy, mediánová filtraci (obr. 4.4), Gaussův filtr (obr. 4.5) a Sobelův operátor (obr. 4.6). Oba jsou převzaty z knihovny VTK, která obaluje původní ITK implementaci.

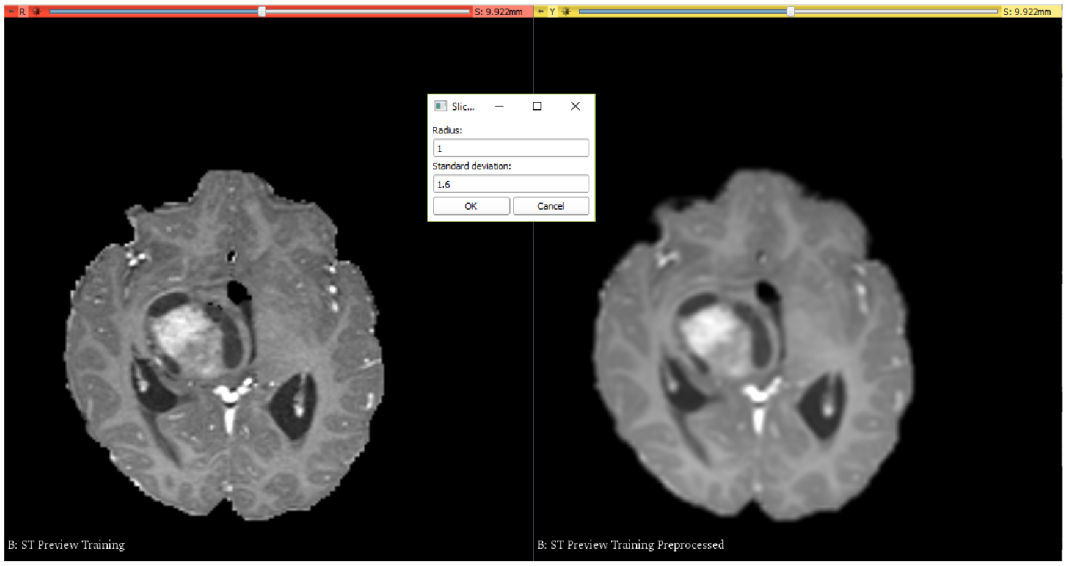
4.5 Klasifikátory

Podobně jako u předzpracovacích algoritmů, i všechny klasifikátory dědí z virtuální třídy. `SupervisedClassifier` obsahuje dvě grafické komponenty, jednu pro učební a druhou pro klasifikační fázi. V odvozených třídách je nutné přetížit metody pro serializaci a deserializaci a metody učební a klasifikační.

Ačkoliv nejsou další požadavky na klasifikátor kladeny, všechny čtyři již implementované algoritmy dále využívají procesů, zjednodušeně zobrazených na obr. 4.7. Implementovány jsou binární klasifikátory N SVM a C SVM knihovny dlib a vícetřídové C SVM a Random Forest knihovny Shark. Tyto dvě knihovny byly vybrány z toho důvodu, že obsahují i další vhodné klasifikační algoritmy a dostatečnou dokumentaci s příklady použití. Při přijetí sady dat z `VolumeManager` se ukazatele na data

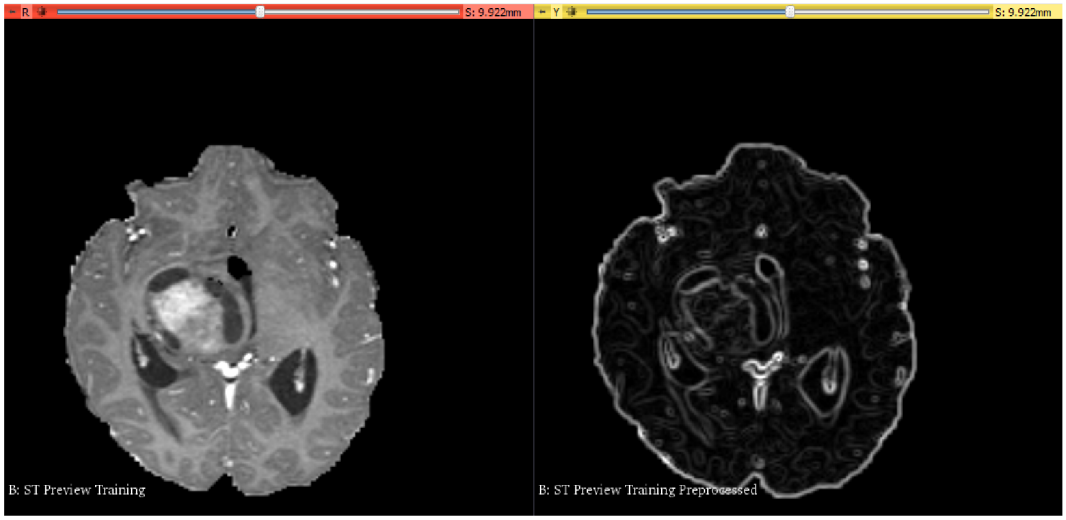


Obr. 4.4: Ukázka předzpracování pomocí mediánové filtrace. Uprostřed nastavovací grafické rozhraní.



Obr. 4.5: Ukázka předzpracování Gaussova filtru.

předávají speciálnímu vláknu pro správu dat klasifikátoru. V první fázi učení se tato data transformují a načítají do určeného kontejneru. Transformace zahrnují normalizaci a změnu označení dle nastavení a potřeb specifického algoritmu. V druhé fázi učení uživatel vybere buď přímé nastavení parametrů učení, nebo prohledání prostoru těchto parametrů a výběr nejlepších. Výběr nejlepší sady parametrů probíhá v optimálním počtu vláken na daném systému, který je získán pomocí Qt. Pětinásobnou křížovou validací se testuje každá sada a metrikou pro optimálnost sady



Obr. 4.6: Ukázka předzpracování pomocí Sobelova operátoru.

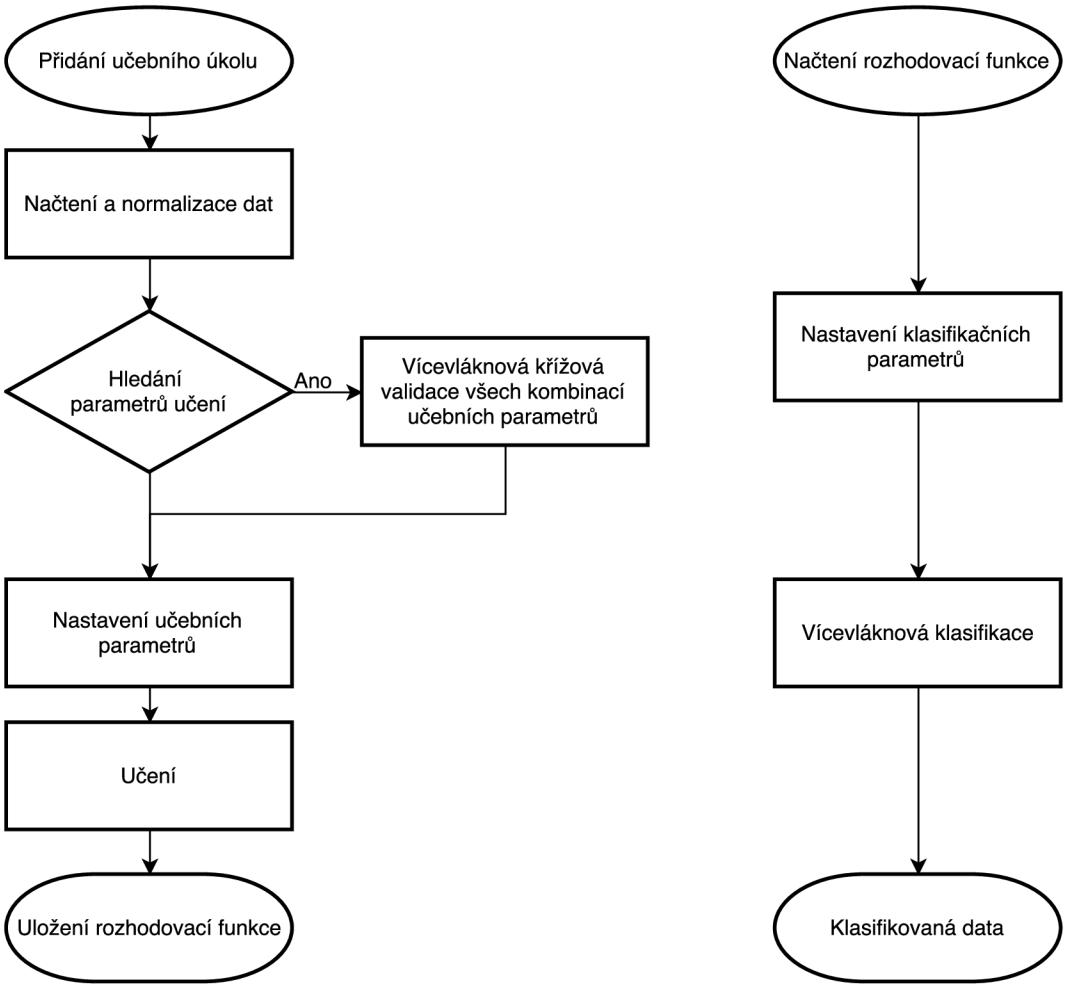
parametrů je součet senzitivity a specificity. Mezivýsledky jsou uloženy v csv souboru pro další zpracování. Následuje naučení klasifikátoru a jeho uložení do souboru spolu s počtem obrazových sad a nastaveným předzpracováním.

Při klasifikaci vlákno pro správu dat klasifikátoru opět přijme ukazatele na MRML uzly, obsahující neoznačená data, a vytvoří optimální počet vláken, která souběžně klasifikují několik řezů.

4.6 Uživatelská příručka

Tato podkapitola slouží k orientaci v uživatelském rozhraní a k vysvětlení principu použití modulu pro případné uživatele. Na obr. 4.8 je zobrazen základní stav uživatelského rozhraní pro učení a na obr. 4.9 pro klasifikaci.

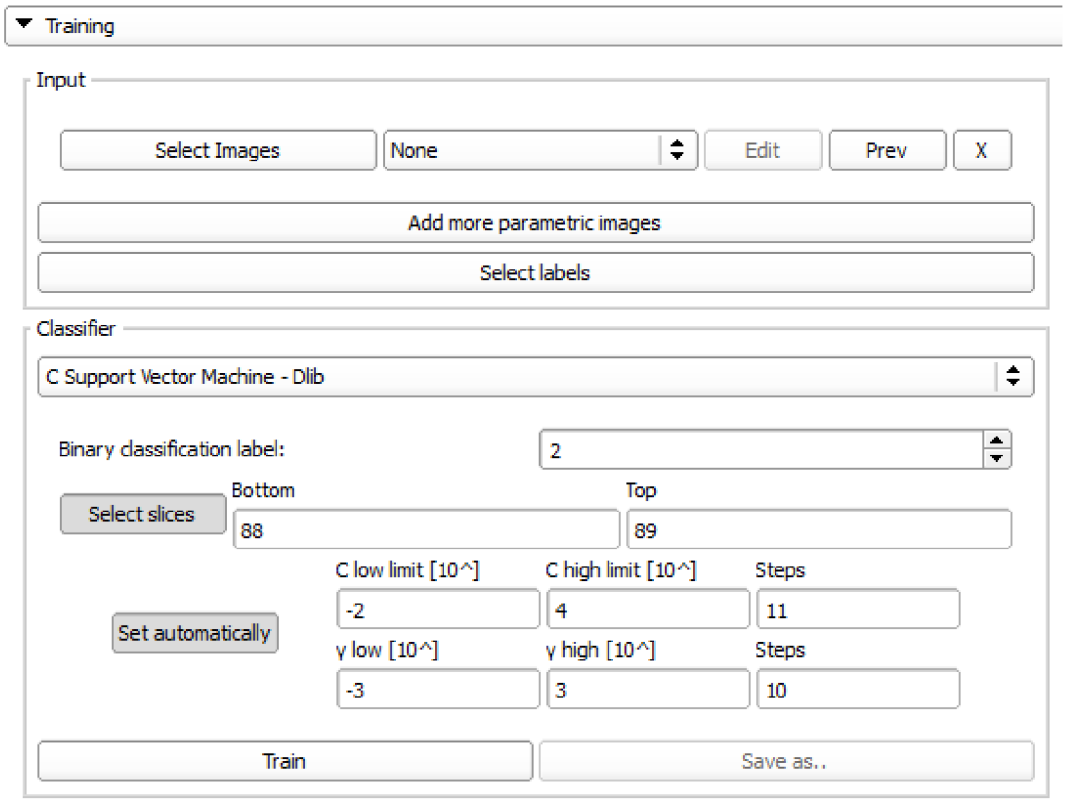
První informace, které je potřeba, aby uživatel zadal, jsou vstupní obrazy. V horní části učebního modulu se nachází řádek s několika tlačítky. První tlačítko „Select Images“ vyvolá dialog pro výběr souborů. Zde uživatel vybere jeden, nebo více obrazů, které budou reprezentovat jednu dimenzi učebního vektoru. V případě výběru více obrazů je potřeba zachovat pořadí při nastavování dalších dimenzí i pravdivostních obrazů, aby si jednotlivé voxely odpovídaly. Druhé tlačítko zleva otevře vyskakovací nabídku předzpracovacích algoritmů. Tlačítko „Edit“ slouží k nastavení předzpracovacího algoritmu, je-li nastavení možné. „Prev“ zobrazuje ukázku předzpracovaného obrazu. Křížkem se odstraňuje celý řádek, tlačítkem „Add more parametric images“ ve spodní části sekce „Input“ naopak přidává. „Select labels“ vybírá pravdivostní segmentované obrazy. Příkladem použití budiž situace, kdy pro učení algoritmu uživatel potřebuje Fluid-attenuated Inversion Recovery (dále



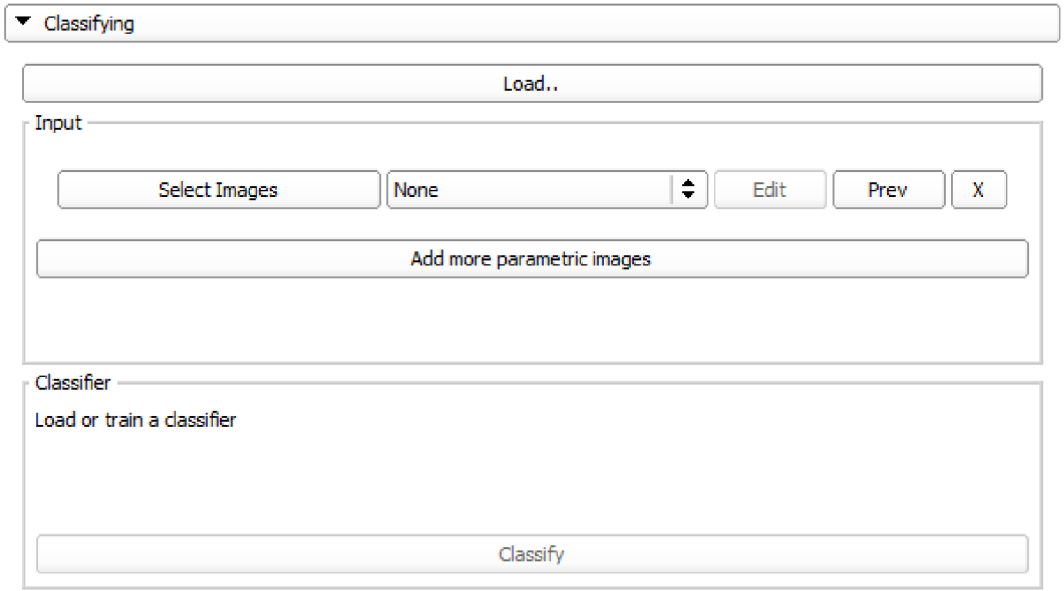
Obr. 4.7: Diagram učení (vlevo) a klasifikace (vpravo).

FLAIR) a T2-vážené obrazy tří pacientů jako dimenze učebních vektorů. FLAIR váhované obrazy navíc potřebují odstranit impulzní šum. Uživatel tedy přidá jeden řádek pro výběr a předzpracování do celkového počtu dvou řádků. Na prvním řádku vybere T2-vážené obrazy v pořadí pacient 1, pacient 2, pacient 3 a zbytek řádku nechá v základním nastavení. Na druhém řádku vybere ve stejném pořadí obrazy získané sekvencí FLAIR a vybere 3D mediánový filtr jako předzpracovací algoritmus. Ve vyskakovacím okně nastavení mediánového filtru uživatel nastaví velikost masky, například tak, že v rovinách nasnímaných řezů nastaví nejednotkovou velikost masky, zatímco ve třetí dimenzi nastaví velikost masky na 1. V tu chvíli používá 2D mediánový filtr a jeho účinnost si snadno ověří tlačítkem „Prev“.

V dalším kroku se uživatel přesune k sekci „Classifier“, kde vybere a nastaví klasifikátor. U všech implementovaných algoritmů lze zadat výběr řezů pro zrychlení učení například při hledání parametrů. U učení binárních algoritmů knihovny dlib lze zadat vybranou hodnotu pravdivostního obrazu. Voxely se všemi ostatními

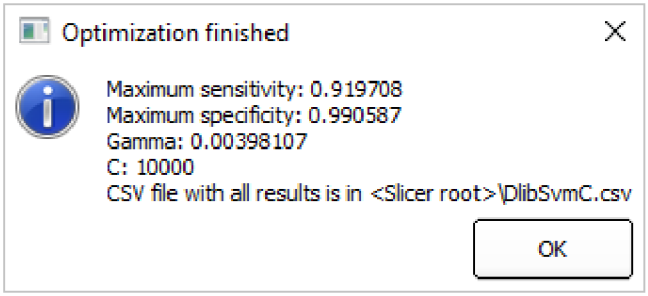


Obr. 4.8: Uživatelské rozhraní pro učení.



Obr. 4.9: Uživatelské rozhraní pro klasifikace.

hodnotami budou považovány za negativní vzorek. U knihovny shark a jejích vícetřídových klasifikátorů lze zadat několik čárkou oddělených hodnot. Dále uživatel nastaví parametry učení buď přímo, nebo je nechá vyhledat automaticky se zadáním vyhledávacího prostoru a stiskne tlačítko „Train“. Po nalezení optimálních parametrů je klasifikátor pomocí těchto naučen a uživateli je zobrazen výsledek vyhledávání (obr. 4.10). Nakonec uživatel klasifikátor i nastavení předzpracování a počtu dimenzí učebních vektorů uloží jako soubor pro další použití.



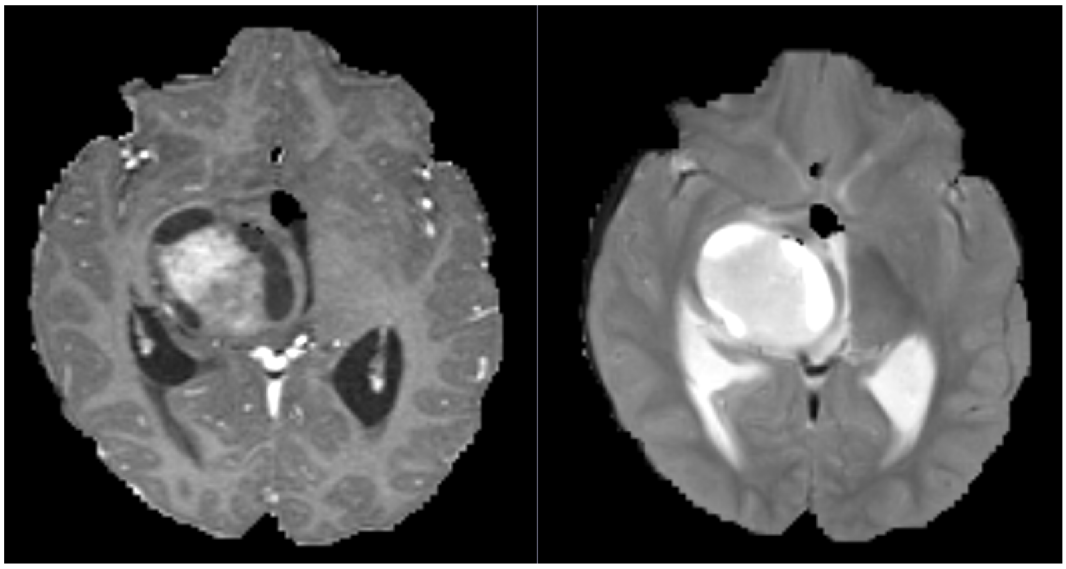
Obr. 4.10: Ukázka výsledku vyhledávání učebních parametrů.

Při klasifikaci uživatel načte soubor klasifikátoru s informacemi o učebních vektorech a jejich předzpracování. Kromě výběru souboru pak uživatel již nemusí v sekci „Input“ nic měnit. Pokud přeci jen změní například nastavení předzpracovacího algoritmu, není zaručena přesná klasifikace a v případě přídání dalších dimenzí (parametrických obrazů) není zaručena ani stabilita programu. Uživatel dále zadá rozsah řezů u kterých chce provést klasifikaci. U binárních algoritmů knihovny dlib ještě uživatel může zadat práh pro diskriminaci tříd. Po kliknutí na tlačítko „Classify“ a doběhnutí klasifikačních vláken se v zobrazovacích oknech objeví výsledné segmentované obrazy.

5 DOSAŽENÉ VÝSLEDKY

Jako testovací obrazy byly použity obrazy s dobře diferencovanými tumory databáze Brain Tumor Segmentation (dále BRATS) z roku 2013.

Ke zhodnocení použitelnosti rozšíření bylo provedeno na každém z implementovaných algoritmů několik testovacích scénářů. Prvním z nich bylo zhodnocení vlivu počtu řezů, na kterých je algoritmus naučen, na senzitivitu a specifictu. Bylo provedeno hledání nejvhodnějších parametrů při každém testu a výsledná senzitivita a specificita byla získána klasifikací objemu dat, kde se nachází tumor (řezy 64 až 104 u použitého obrazu). Tento test byl proveden na obrazech LG_0001 a postupné rozšiřování objemu dat probíhalo kolem středu tumoru, tedy na řezu 88 axiální roviny (obr. 5.1).



Obr. 5.1: Řez 88 LG_0001_T1C (vlevo) a LG_0001_T2

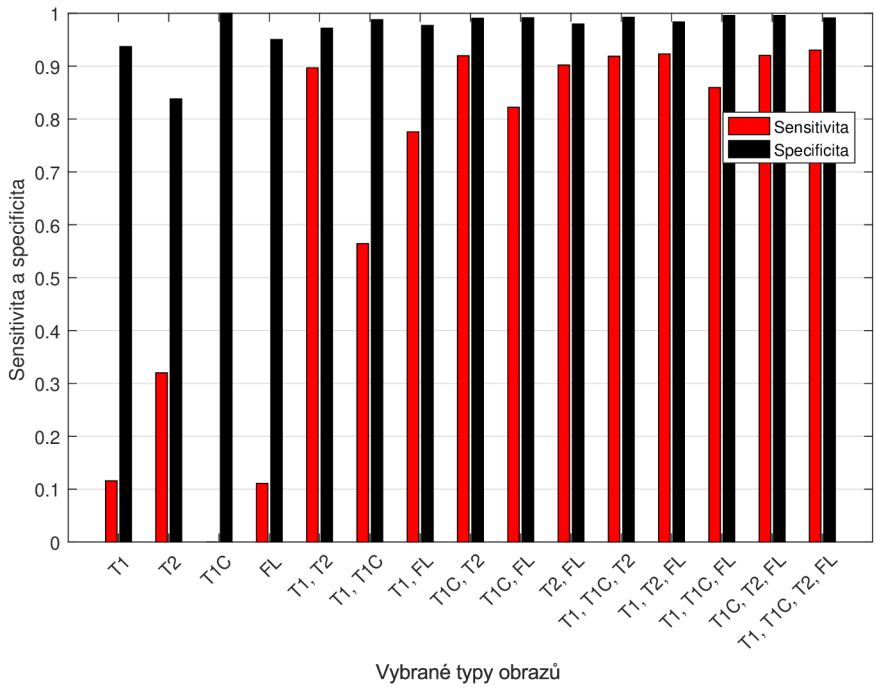
Druhým testem bylo vyzkoušení všech kombinací typů dostupných obrazů z magnetické rezonance, jmenovitě T1-vážené s kontrastem i bez něj, T2-vážené a obrazy získané sekvencí FLAIR. Z časových důvodů test probíhal na jednom řezu obrazů LG_0001, konrétně řezu 88 axiální roviny. Sensitivita a specificita byla získána pětinásobnou křížovou validací vstupních dat. Data tedy byla náhodně rozdělena na pět dílů a pro každý díl byl algoritmus naučen, na zbylých čtyřech dílech byl pak testován.

Poslední test probíhal na obrazech LG_0001_T1C a LG_0001_T2, tentokrát na celém objemu. Cílem bylo zjistit vliv parametrů učení na specificitu a sensitivitu. Ta byla získána opět pětinásobnou křížovou validací. Z tohoto testu byla poté vzata

nejlepší sada sensitivity a specificity a použita k finálnímu srovnání algoritmů. Ve finálním srovnání jsou také časové a prostorové nároky všech algoritmů.

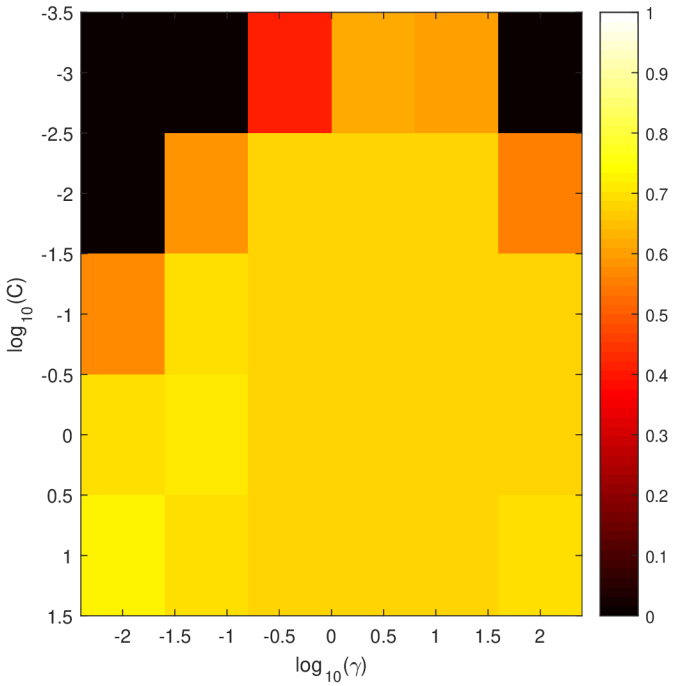
5.1 C SVM - Dlib

Prvním z testovaných algoritmů byl C SVM knihovny dlib. Verze knihovny shark testována nebyla, jelikož obě knihovny využívají totožnou implementaci a rozdíly mezi nimi by byly minimální. Na obr. 5.2 lze vidět, že tento algoritmus dosahuje přesnosti přes 0,9 již s obyčejnými T1- a T2-váženými obrazy, jejichž hodnoty jasu jsou použity jako dimenze učebních vektorů. Nejlepší z dvouobrazových testů se zdá být kombinace kontrastního T1-váženého a T2-váženého. Některé víceobrazové testy sice dosahují i vyšších hodnot senzitivity a specificity, ale jejich přínos je vzhledem k delším výpočtovým časům diskutabilní.

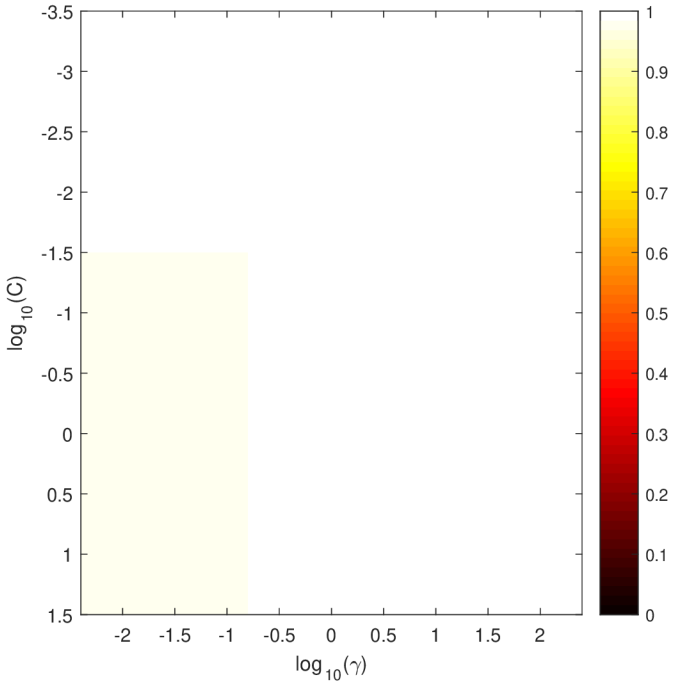


Obr. 5.2: C SVM - vliv vybraných typů obrazů na senzitivitu a specificitu

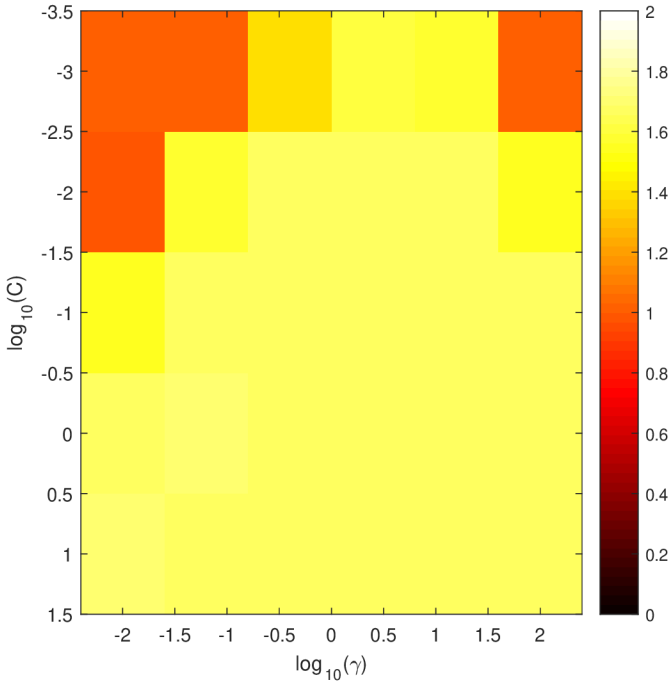
Při prohledávání prostoru parametrů na intervalech $\langle 0,0003; 10\rangle$ pro C a $\langle 0,01; 100)$ pro parametr γ nebyly pozorovány výrazné vlivy na specificitu. Pro senzitivitu se jeví nízké γ a relativně vysoké C jako nejvhodnější.



Obr. 5.3: C SVM - hledání parametrů, sensitivita

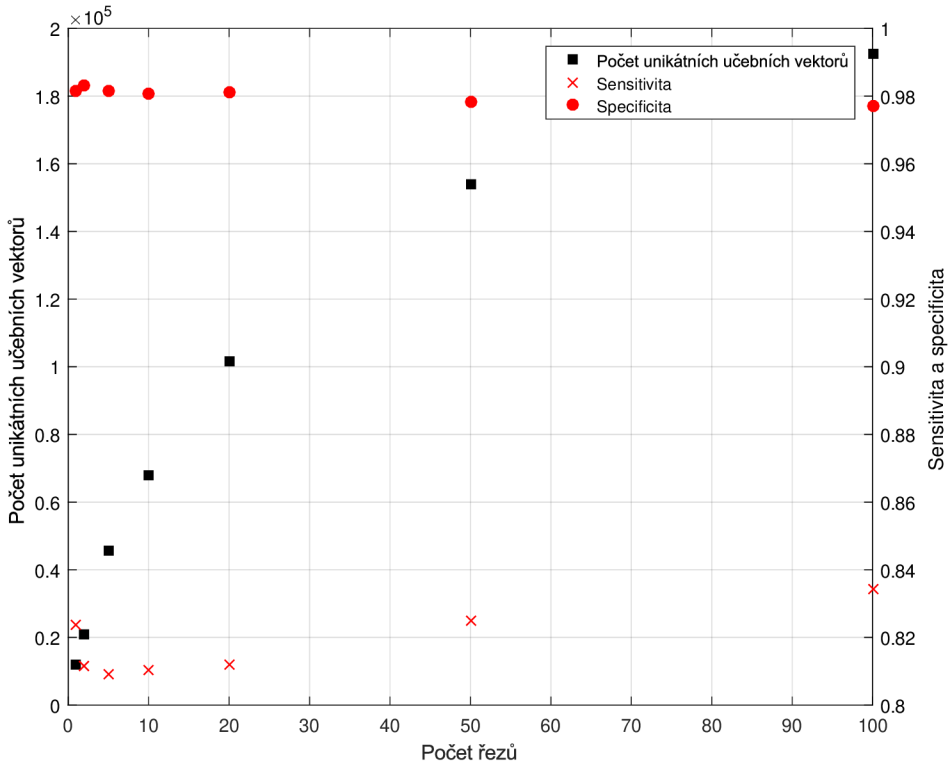


Obr. 5.4: C SVM - hledání parametrů, specificita



Obr. 5.5: C SVM - hledání parametrů, součet sensitivity a specificity

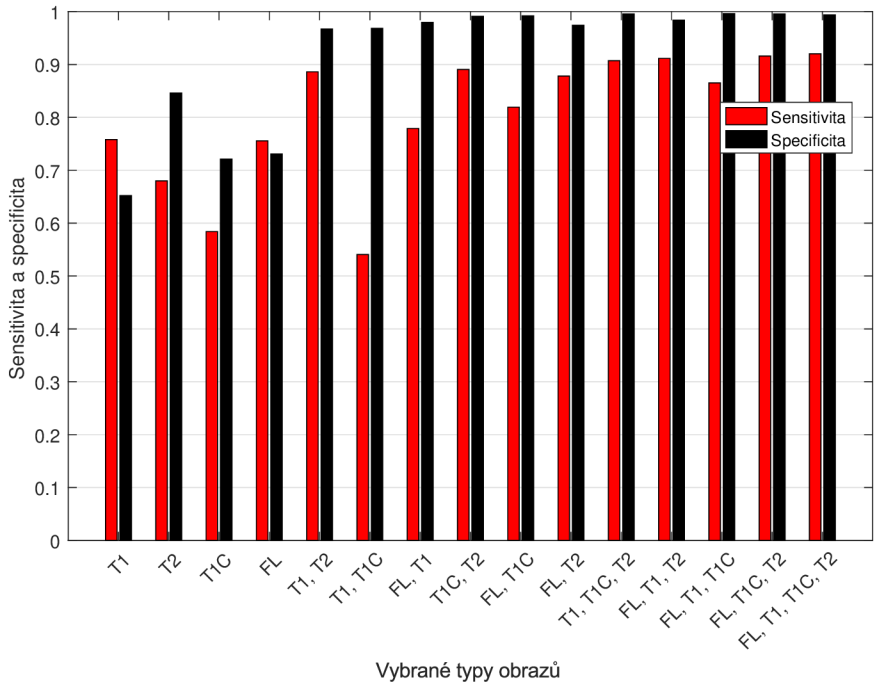
Počet řezů má minimální vliv na sensitivitu a specificitu při klasifikaci tumorové oblasti testovacího obrazu, jak dokazuje obr. 5.6. Nad počet 50 řezů lze pozorovat mírný nárůst sensitivity, tedy poměru správně nalezených tumorových voxelů vůči všem klasifikovaným tumorovým voxelům a ještě mírnější pokles specificity, tedy poměr správně označených negativních vzorků vůči všem negativně klasifikovaným.



Obr. 5.6: *C* SVM - vliv počtu řezů na sensitivitu, specificitu a počet učebních vektorů

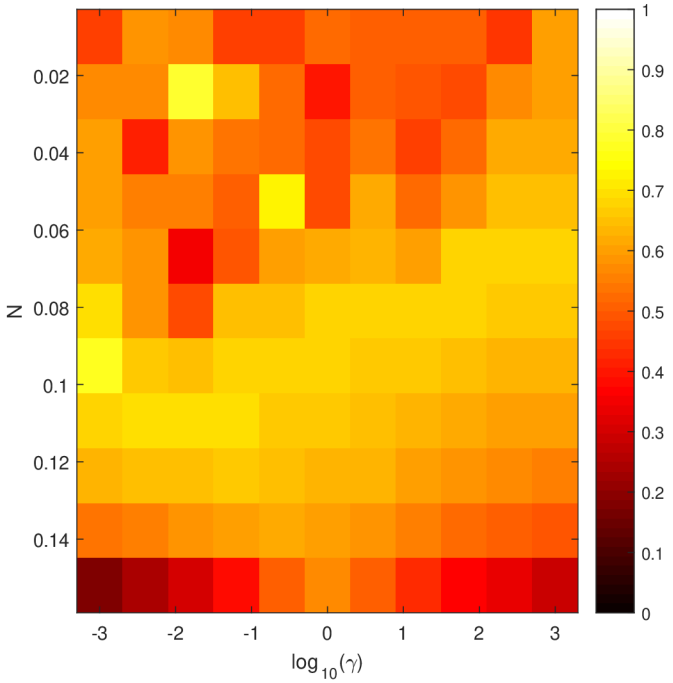
5.2 Nu SVM - Dlib

N SVM knihovny Dlib, označený jako Nu SVM, dosahuje použitelných hodnot sensitivity a specificity pomocí kombinace obrazů T1 a T2, případně T1C a T2. Drobného zlepšení lze dosáhnout kombinací T1C, T2 a FLAIR obrazu, jak je zaznamenáno na obr. 5.7.

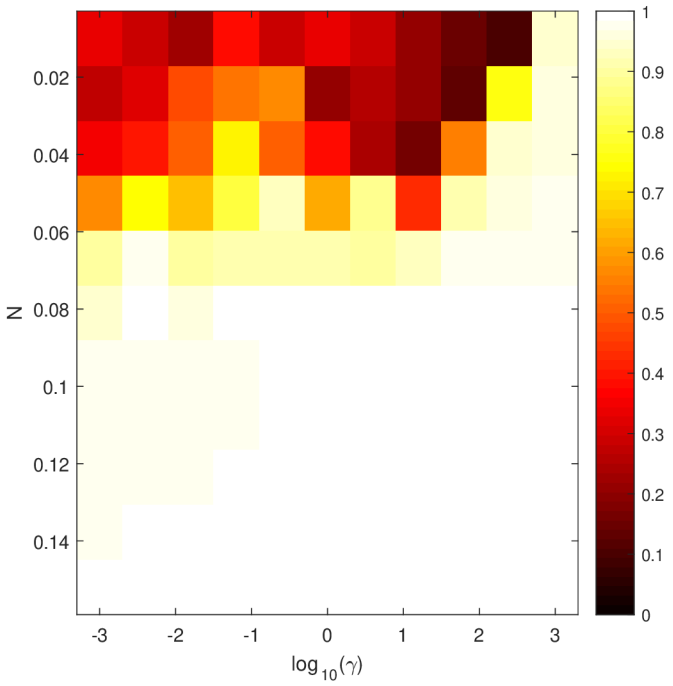


Obr. 5.7: N SVM - vliv vybraných typů obrazů na senzitivitu a specificitu

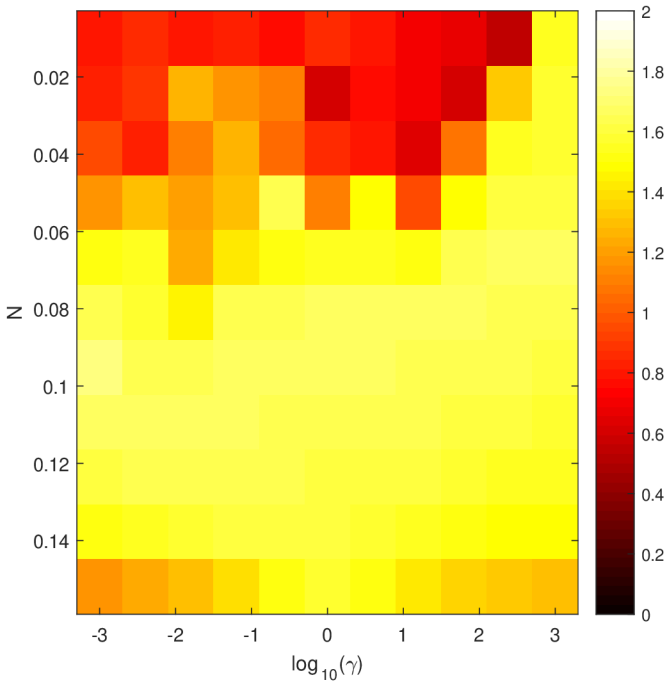
Prohledávání prostoru parametrů probíhalo na intervalu $\langle 0,01; 0,16)$ pro N a $\langle 0,001; 1000)$ pro parametr γ. Velmi nízké hodnoty N mají negativní efekt na specificitu (obr. 5.9) a velmi vysoké na sensitivitu (obr. 5.8). Parametr γ se zdá být vhodnější volit vyšší, ovšem kombinace parametrů s největším součtem sensitivity a specificity je pro $\gamma = 0,003$ a $N = 0,095$, jak je viditelné na obr. 5.10. Obecně klasifikátor není příliš striktní na výběr parametrů a dobrých výsledků dosahoval ve velkém procentu kombinací. Je také na místě poznamenat, že vysoké hodnoty γ mají negativní dopad na čas výpočtu.



Obr. 5.8: N SVM - hledání parametrů, sensitivita

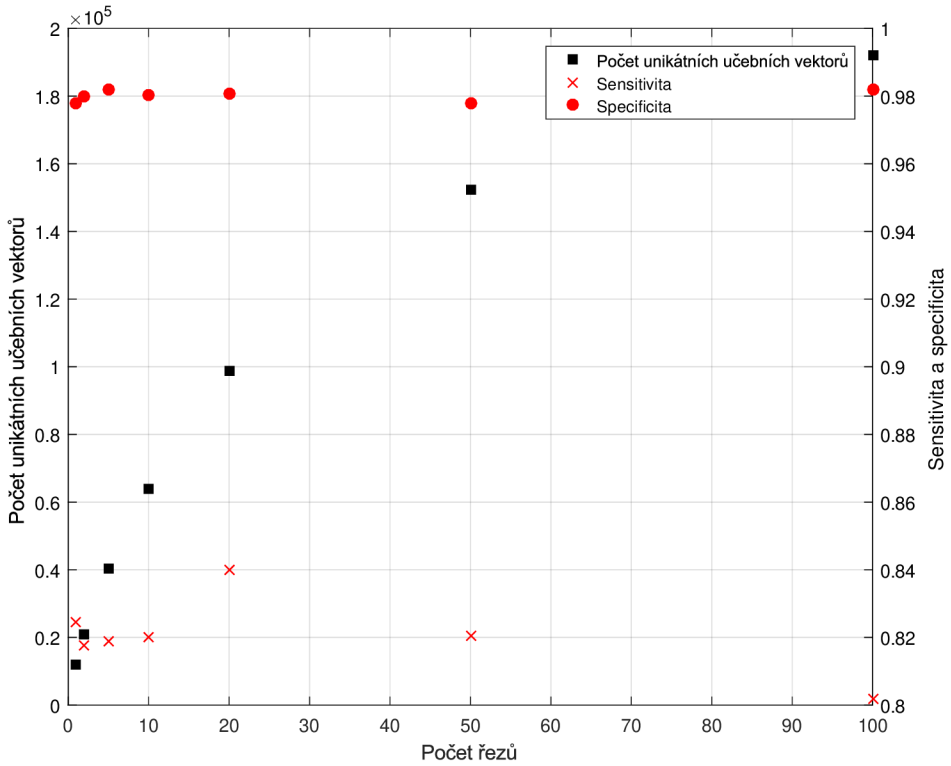


Obr. 5.9: N SVM - hledání parametrů, specificita



Obr. 5.10: N SVM - hledání parametrů, součet sensitivity a specificity

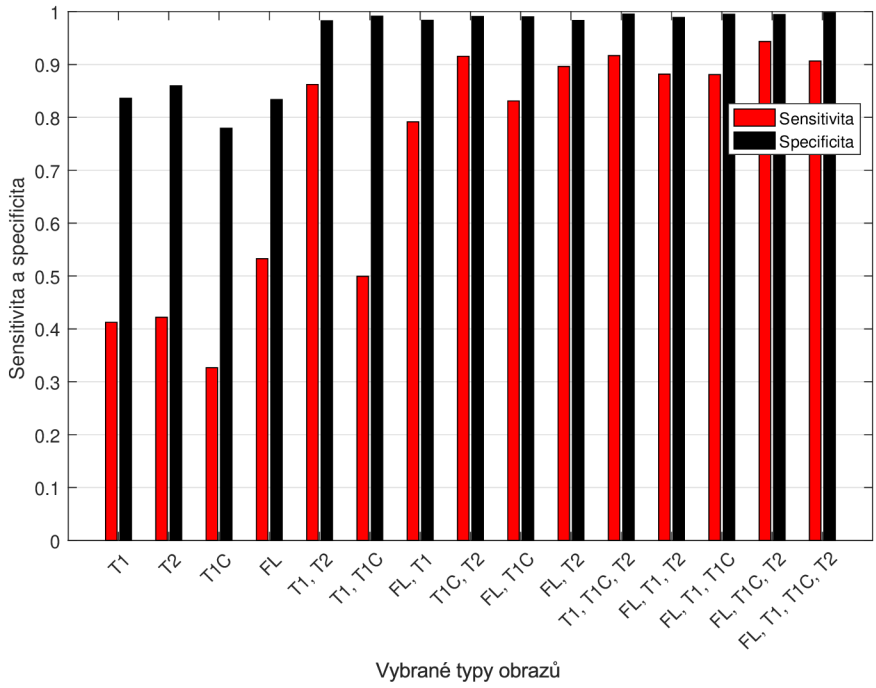
Počet řezů, respektive počet učebních vektorů má minimální vliv na sensitivitu a specificitu klasifikace tumorové části testovaného obrazu, jak vyznačuje obr. 5.11



Obr. 5.11: N SVM - vliv počtu řezů na sensitivitu, specificitu a počet učebních vektorů

5.3 Náhodný les - Shark

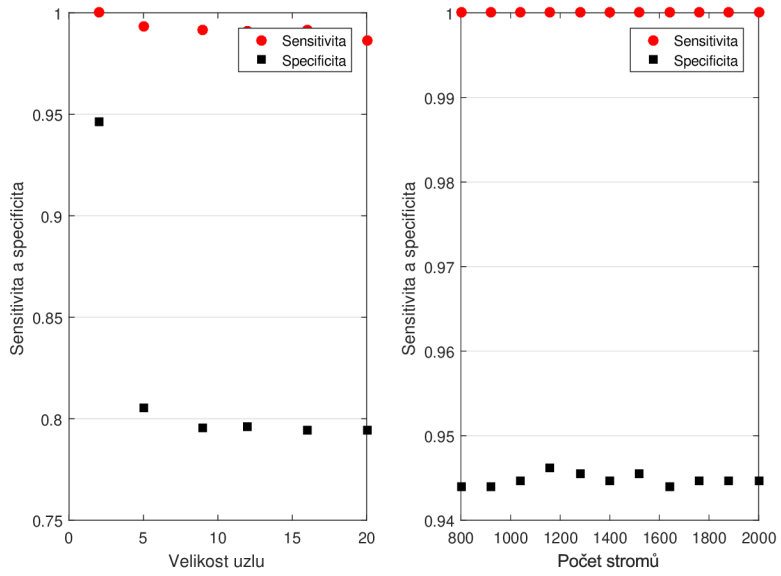
Pro náhodný les se jeví jako nejvhodnější kombinace obrazů kontrastní T1- a T2-vážené. Opět je nejlepší kombinací FLAIR, T1C a T2, ovšem rozdíl s druhým nejlepším výsledkem minimální. Všechny výsledky jsou zobrazeny v grafu na obr. 5.12



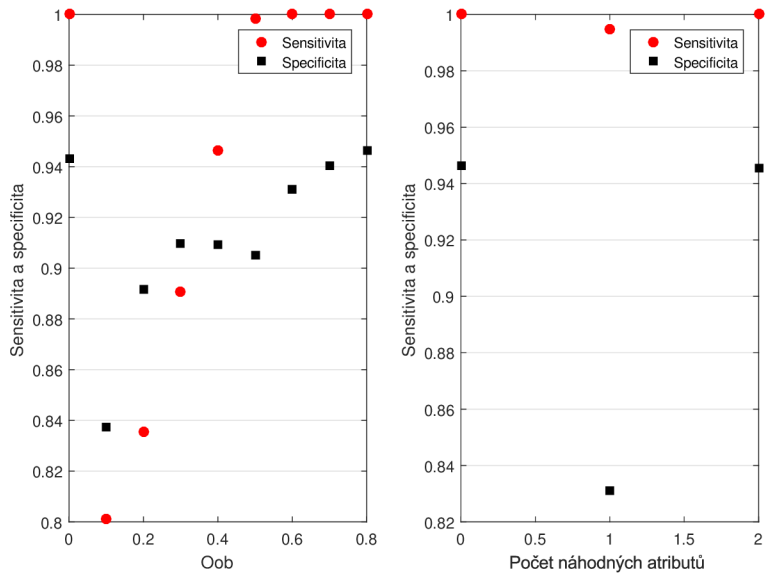
Obr. 5.12: Náhodný les - vliv vybraných typů obrazů na senzitivitu a specificitu

Z obr. 5.13 a 5.14 lze usuzovat, že celkový počet stromů nad 800 již nemá vliv na přesnost algoritmu. Dále je patrné, že ideální je menší množství velikosti uzlů, což vede obecně k jednodušším rozhodovacím stromům. Závěr ohledně vlivu procenta učebních vzorků pro jednotlivé stromy, které nepochází z původního rozdělení (tzv. bootstrappingu) je možné učinit takový, že ideální hodnota je nula, případně co nejvyšší. Podobně lze shrnout i výsledky týkající se posledního atributu učení náhodného lesu, počtu náhodných atributů.

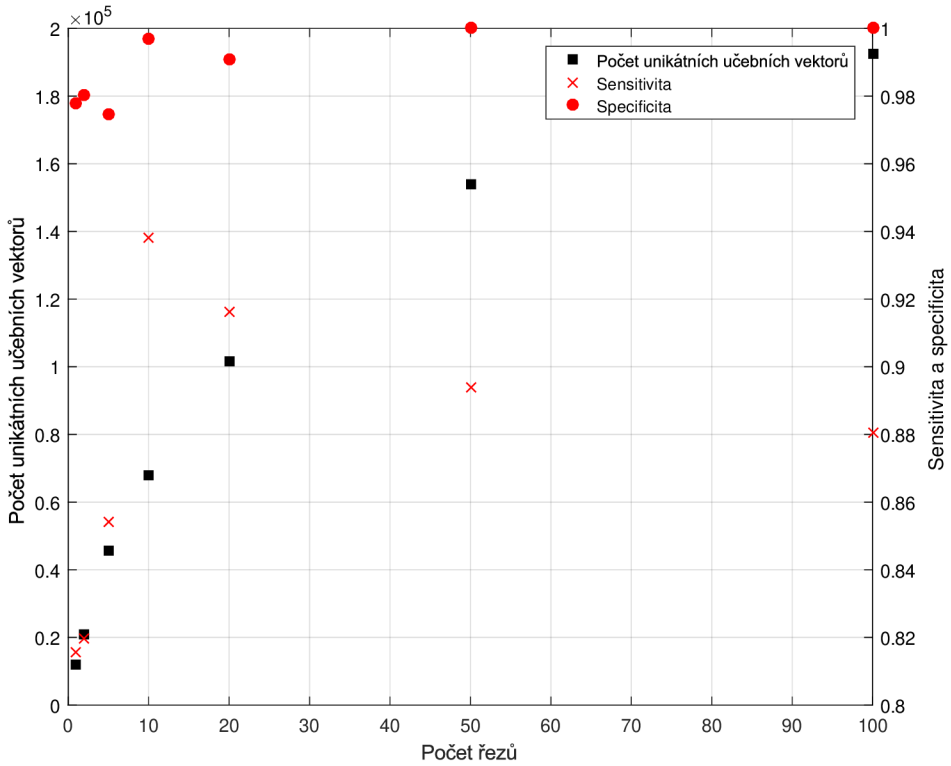
Náhodný les dle obr. 5.15 při přidávání počtu řezů k učení vykazuje minimální zlepšení. Jako jediný z testovaných algoritmů dosáhl maximální specificity na učebních datech.



Obr. 5.13: Náhodný les - hledání parametrů, vlevo vliv velikosti uzlů a počtu stromů



Obr. 5.14: Náhodný les - hledání parametrů, vlevo vliv procenta vzorků mimo učební podmnožinu jednotlivých stromů (Oob) a počtu náhodných atributů v uzlech



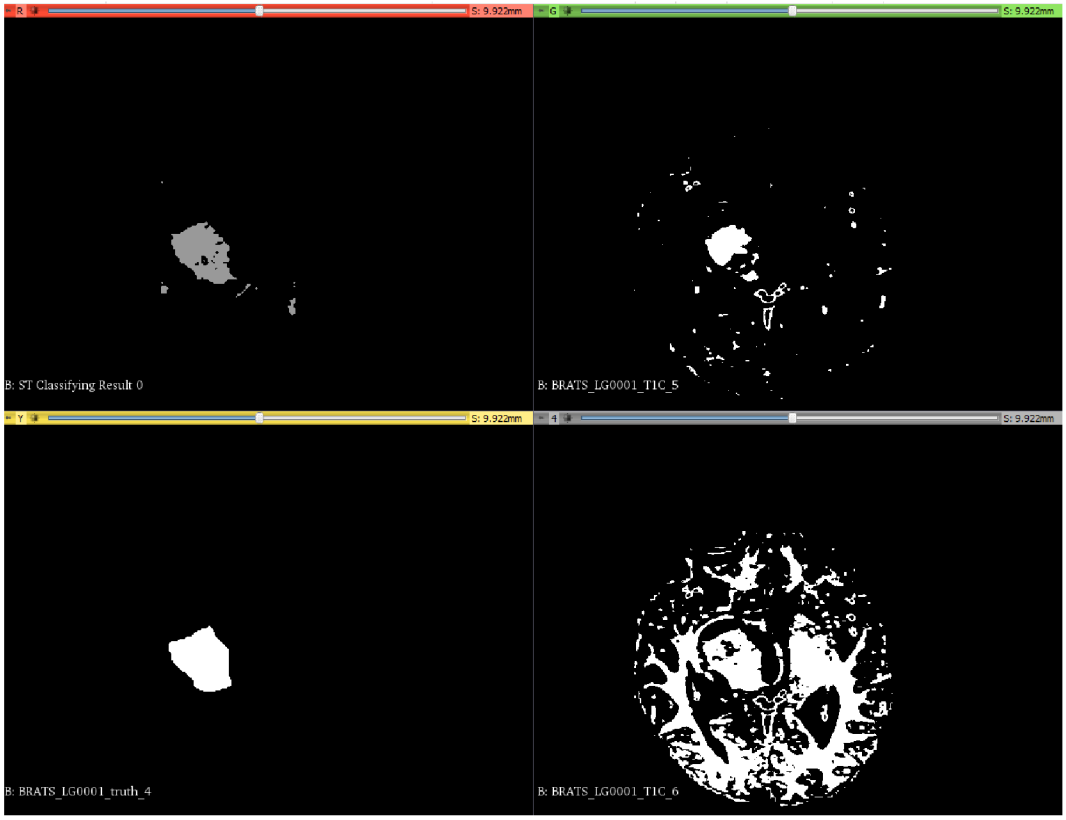
Obr. 5.15: Náhodný les - vliv počtu řezů na sensitivitu, specificitu a počet učebních vektorů

5.4 Srovnání klasifikátorů

Ke srovnání algoritmů byla vybrána maximální hodnota sensitivity a specificity z testů na výběr parametrů. Navíc byla přidána nejlepší hodnota získatelná dvojitým prahováním T1C obrazu. Časové a prostorové srovnání probíhalo na operačním systému Windows 10 Home verze 1703 s 8 GB DDR3 RAM a procesorem intel i7-6700HQ. Úkolem bylo učení s nejlepší kombinací parametrů na celém testovacím obrazu a následná klasifikace.

Tab. 5.1: Srovnání dosažených maximálních sensitivit a specificit

	Maximální sensitivita	Maximální specificita
C SVM - Dlib	0,7196	0,9813
N SVM - Dlib	0,7710	0,9706
Náhodný les - Shark	1,0000	0,9462
Dvojité prahování	0,9659	0,8453



Obr. 5.16: Srovnání výsledků N SVM a dvojitého prahování. Vlevo nahoře výsledek segmentace pomocí N SVM, vlevo dole pravdivostní obraz. Vpravo nahoře ručně dvojitě prahovaný obraz. Vpravo dole dvojitě prahovaný obraz s nejlepším číselným výsledkem součtu sensitivity a specificity.

Tab. 5.2: Srovnání časových a prostorových náročností algoritmů

	Čas výpočtu učení [s]	Čas výpočtu segmentace [s]
C SVM - Dlib	992	961
N SVM - Dlib	691	1000
Náhodný les - Shark	778	1240
	RAM při učení [MB]	RAM při segmentaci [MB]
C SVM - Dlib	283	57
N SVM - Dlib	270	45
Náhodný les - Shark	570	47

6 DISKUZE

Dosažené výsledky jsou ve shodě, případně o něco vyšší s již dostupnými analýzami v literatuře. Výzkum pro použití SVM ke klasifikaci pomocí kontrastních T1-vážených obrazů dosahuje přesnosti, tedy poměru správně klasifikovaných vektorů vůči všem, 87.5%[26]. Jiný přístup k úloze byl zvolen v [27], kde rozlišují pouze abnormální a normální řezy. Vstupem do vektoru je zde výsledek fuzzy c-means analýzy T1- a T2-vážených obrazů. Tímto způsobem byla dosažena přesnost 92.71%. Srovnáním různých metod segmentace se zabýval Sumithra et al [28], kde z testovaných dosahoval SVM nejlepších výsledků se specificitou přes 80% a sensitivitou okolo 85%. Lepší hodnoty získáné v této práci lze odůvodnit testováním na relativně omezeném vzorku dat, k dodatečnému zpřesnění hodnocení využitelnosti jednotlivých algoritmů by bylo vhodné je také křížově otestovat na jiných obrazech, než byl testovací. Získané výsledky a citovaná literatura potvrzují vhodnost strojového učení k automatizované klasifikaci.

Naprogramované rozšíření splnilo všechny vytyčené cíle. Optimálně využívá systémové prostředky a je stabilní. Drobná vylepšení by byla možná v oblasti načítání obrazů, kde současný postup vyžaduje, aby soubory, které definují jednu dimenzi učebních vektorů, byly v jedné složce. Možné lepší, případně dodatečné řešení by bylo k načítání souborů využít rozhraní 3D Slicer a v rozšíření pouze vybírat načtené MRML uzly, jak je zvykem v ostatních rozšířeních. Dalším hypotetickým nedostatkem by mohla být serializace v binárně čitelném souboru. Zde by z důvodu zpětné kompatibility a jasné definice bylo vhodnější použít jazyk XML i za cenu větší velikosti konečného souboru. Při testování bylo zjištěno, že serializační funkce knihovny shark pro algoritmus náhodný les využívá člověkem čitelný způsob zápisu. Tento způsob serializace má za následek rychlé narůstání velikosti souboru v závislosti na počtu stromů. Nad velikost několika set stromů již naráží na indexační limitace třídy `QByteArray` knihovny Qt, která je použita jako mezičlánek serializačních funkcí knihovny shark a uložením do souboru. Při testování nebylo spatřeno zpřesnění klasifikace při použití více než dvou set stromů, ovšem i tehdy může serializovaný soubor dosahovat velikosti několika stovek MB na pevném disku v závislosti na velikosti stromu. Náhodný les je také algoritmem ideálním k hybridní paralelizaci s použitím grafické karty[29].

Funkcionalitu modulu by bylo vhodné doplnit o možnost řetězení předzpracovacích algoritmů, ovšem otázkou zůstává, zda se konceptuálně do rozšíření vůbec hodí komplexní předzpracování, když 3D Slicer již tyto algoritmy obsahuje. Klasifikační část modulu by se dala doplnit o možnost souběžně klasifikovat obraz více klasifikátory a výsledek by si mohl uživatel zvolit například jako logický součet, či součin výsledků z jednotlivých klasifikátorů.

ZÁVĚR

Byly popsány běžně používané klasické metody segmentace tomografických dat. U vybraných metod byl popsán aktuální vývoj a použití v medicíně. Dále byly předvedeny používané a experimentální klasifikační přístupy strojového učení. Byla provedena rešerše dostupných knihoven strojového učení v jazyce C++. Byl popsán program 3D Slicer, jeho programová architektura a možnosti rozšíření.

V dalších kapitolách byly popsány cíle rozšíření pro segmentaci tomografických dat. Programová architektura byla podrobně vysvětlena spolu s popisem použitých knihoven strojového učení. Z těchto byly použity algoritmy C SVM (knihovna dlib i shark), N SVM (knihovna dlib) a náhodný les (knihovna shark). Práce obsahuje uživatelskou příručku pro případného uživatele rozšíření.

Dále byl objasněn způsob testování použitelnosti rozšíření a vybraných implementovaných algoritmů strojového učení. Tři vybrané klasifikátory, jmenovitě N SVM a C SVM knihovny dlib a náhodný les knihovny shark, byly testovány. U každého z těchto klasifikátorů byl sledován vliv typu vstupních obrazů na maximální dosaženou sensitivitu a specificitu. Ze všech kombinací T1-vážených obrazů před i po aplikaci kontrastní látky, T2-vážených a obrazů získaných sekvencí FLAIR je nutné použít minimálně sady dvou obrazů k dosažení interpretovatelného výsledku. Z těchto sad se jeví kombinace kontrastního T1-váženého a T2-váženého obrazu jako dobrý výchozí bod pro všechny testované klasifikátory. V případě kontraindikace kontrastní látky je možné také použít v kombinaci s T2-váženým T1-vážený obraz v případě klasifikátorů SVM, nebo FLAIR v případě použití náhodného lesu. Použití více obrazů jako dimenzí učebních vektorů přináší minimální zlepšení sensitivit i specificit za cenu značného prodloužení výpočetních časů.

Prohledávání prostoru parametrů každého klasifikátoru, za použití T2-váženého a kontrastního T1-váženého obrazu, lze shrnout do několika doporučení pro výběr počátečních bodů vlastního nastavení klasifikátorů. U N SVM se zdá být vhodné volit parametr N v rozmezí $0,08$ a $0,14$, zatímco parametr γ v testovaném intervalu neměl na přesnost klasifikace vliv. Z teoretických předpokladů lze odhadovat, že nižší hodnoty N by měly vést k přesnějším výsledkům, za současného zvýšení rizika přeučení algoritmu na specifické hodnoty učebních vektorů. Omezenější prohledání prostoru parametrů klasifikátoru C SVM ukázalo, že je vhodnější volit $\gamma << 1$ a relativně vysoké hodnoty $C > 10$. Vzhledem k časovým nárokům C SVM nebyly prozkoumány vyšší hodnoty C. Klasifikátor náhodný les je vhodné dle získaných výsledků nastavovat s co nejjednoduššími uzly, počtu stromů do několika set, pouze s vzorky z učebních podmnožin a bez náhodných atributů.

U dalšího z provedených testů bylo dokázáno, že algoritmy SVM benefitují při dané úloze ze zvýšení počtu učebních řezů minimálně, pokud jsou ve vybraných

řezech dostatečně zastoupeny klasifikované třídy. Klasifikátor využívající náhodný les dokázal využít přidaná data o něco lépe, s rozdílem ve specifitě kolem 10%.

Výsledky lze shrnout úspěchem využitelnosti rozšíření pro danou úlohu. Jako nejpřesnější a nejrychlejší algoritmus se jeví náhodný les a také N SVM. Výhoda N SVM spočívá ve snadné interpretaci parametru N. Výsledky byly konzistentní s aktuální literaturou. Nutno dodat, že ač všechno testování probíhalo na několika typech tomografických obrazů magnetické rezonance, navrhované metody strojového učení, ani vytvořené rozšíření uživatele neomezují ve výběru jiných typů obrazů z jiných snímacích modalit.

Nakonec byla diskutována možná rozšíření funkcionality modulu. Zdrojový kód modulu společně se zkompilovanými dynamicky linkovanými knihovnami, vhodnými pro použití ve 3D Slicer verze 4.6.0 jsou umísteny na repozitáři na serveru GitHub[30].

LITERATURA

[1] JAN, J. *Číslicová filtrace, analýza a restaurace signálů.* 2. upr. a rozš. vyd. Brno: VUTIUM, 2002. ISBN 80-214-1558-4.

[2] SONKA, M., HLAVAC, V., BOYLE, R. *Image processing, analysis, and machine vision. 3rd ed., International student edition.* Thomson, 2008 [cit. 4.12.2016] ISBN: 0495244384

[3] SANTHI, K., WAHIDA BANU, R.S.D. *Adaptive contrast enhancement using modified histogram equalization* [online] 2015, Optik - International Journal for Light and Electron Optics, 126(19), 1809-1814 [cit. 1.12.2016] DOI: 10.1016/j.ijleo.2015.05.023. ISSN: 00304026 Dostupné z URL: <`http://linkinghub.elsevier.com/retrieve/pii/S0030402615003198`>

[4] CHANG, D.C., WU, W.R. *Image contrast enhancement based on a histogram transformation of local standard deviation* [online] IEEE Transactions on Medical Imaging . 17(4), 518-531 [cit. 1.12.2016] DOI: 10.1109/42.730397 ISSN: 02780062 Dostupné z URL: <`http://ieeexplore.ieee.org/document/730397/`>

[5] GUPTA, B.,TIWARI, M. *Minimum mean brightness error contrast enhancement of color images using adaptive gamma correction with color preserving framework.* [online] 2016, Optik - International Journal for Light and Electron Optics, 127(4), 1671-1676 [cit. 5.12.2016] DOI: 10.1016/j.ijleo.2015.10.068. ISSN 00304026. Dostupné z URL: <`http://linkinghub.elsevier.com/retrieve/pii/S0030402615014230`>

[6] TIWARI, M., GUPTA, B. *Brightness preserving contrast enhancement of medical images using adaptive gamma correction and homomorphic filtering* [online] 2016, 2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), Bhopal, pp. 1-4. [cit. 5.12.2016] DOI: 10.1109/SCEECS.2016.7509287 Dostupné z URL: <`http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7509287&isnumber=7509260`>

[7] JAYANTHI, V.S., BASKAR, D., PRIYANKA, J.D. *Extraction of myocardial wall from cardiac CT images*[online] 2015, 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). IEEE, s. 1-8 [cit. 5.12.2016] DOI: 10.1109/ICIIECS.2015.7192859. ISBN 978-1-4799-6817-6 Dostupné z URL: <`http://ieeexplore.ieee.org/document/7192859/`>

[8] SUN, H., MA, Y., WU, H., et al. *An improved OTSU's method for CT image boundary contour extraction*[online] 2016 IEEE International Conference on Imaging Systems and Techniques (IST). IEEE, 2016, s. 493-497 [cit. 5.12.2016]. DOI: 10.1109/IST.2016.7738276. ISBN 978-1-5090-1817-8. Dostupné z URL: <http://ieeexplore.ieee.org/document/7738276/>

[9] VAPNIK, V. *Estimation of Dependences Based on Empirical Data* 2006, Springer New York [cit. 7.12.2016] ISBN 978-0-387-30865-4.

[10] WILLIAMS, A. *C++ concurrency in action: practical multithreading* Shelter Island, NY: Manning, c2012. [cit. 7.12.2016] ISBN 1933988770

[11] MISITI, J. *Awesome Machine Learning* [online] [cit. 25.11.2016] Dostupné z URL: <https://github.com/josephmisiti/awesome-machine-learning>

[12] KING, E. D. *Dlib-ml: A Machine Learning Toolkit* [online]. 2009, Journal of Machine Learning Research, 1755-1758, 10 [cit. 25.11.2016] Dostupné z URL: <http://jmlr.csail.mit.edu/papers/volume10/king09a/king09a.pdf>

[13] CURTIN, R. R., CLINE, J. R., SLAGLE, Neil P., et al. *mlpack: A Scalable C++ Machine Learning Library* [online]. 2013, Journal of Machine Learning Research, 801-805, 14 [cit. 25.11.2016] Dostupné z URL: <http://mlpack.org/papers/mlpack2013.pdf>

[14] IGEL, CH., HEIDRICH-MEISNER, V., GLASSMACHERS, T. *Shark* [online]. 2008, Journal of Machine Learning Research, 993-996, 9 [cit. 25.11.2016] Dostupné z URL: <http://www.jmlr.org/papers/volume9/igel08a/igel08a.pdf>

[15] YANGQING, J., SHELHAMER, E., DONAHUE, J., et al. *Caffe: Convolutional Architecture for Fast Feature Embedding* [online]. 2014, arXiv preprint arXiv:1408.5093 [cit. 25.11.2016] Dostupné z URL: <http://ucb-icsi-vision-group.github.io/caffe-paper/caffe.pdf>

[16] MORDVINTSEV, A., OLAH, CH., TYKA, M. *Inceptionism: Going Deeper into Neural Networks* Wednesday, June 17, 2015 [cit. 26.11.2016] Dostupné z URL: <https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>

[17] FEDOROV, A., BEICHEL, R., KALPATHY-CRAMER, J., et al. *3D Slicer as an image computing platform for the Quantitative Imaging Network. Magnetic Resonance Imaging* [online]. 2012, 30(9), 1323-1341 [cit.

23.11.2016]. DOI: 10.1016/j.mri.2012.05.001. ISSN 0730725x. Dostupné z URL: <http://linkinghub.elsevier.com/retrieve/pii/S0730725X12001816> Web: <http://www.slicer.org>

[18] HABERMAN, J. *The overhead of abstraction in C/C++ vs. Python/Ruby* [online]. [cit. 24.11.2016]. Dostupné z URL: <http://blog.reverberate.org/2014/10/the-overhead-of-abstraction-in-cc-vs.html>

[19] GUOY, I. *The Computer Language Benchmarks Game* [online]. [cit. 24.11.2016]. Dostupné z URL: <http://benchmarksgame.alioth.debian.org/u64q/compare.php?lang=python3&lang2=gpp> Web: <http://benchmarksgame.alioth.debian.org/>

[20] SALUCCI, L., BONETTA, D., BINDER, W. *Efficient Embedding of Dynamic Languages in Big-Data Analytics* [online]. 2016, IEEE 36th International Conference on Distributed Computing Systems Workshops (ICDCSW), Nara, Japan, pp. 19-24. [cit. 24.11.2016]. DOI: 10.1109/ICDCSW.2016.40. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7756203&isnumber=7756178>

[21] FILLION-ROBIN, J. C. *qt-easy-build* [online] [cit. 9.12.2016] Dostupné z URL: <https://github.com/jcfr/qt-easy-build/tree/4.8.7>

[22] BREIMAN, L. *Machine Learning* [online]. 45(1), 5-32 [cit. 12.12.2016]. DOI: 10.1023/A:1010933404324. ISSN 08856125. Dostupné z URL: <http://link.springer.com/10.1023/A:1010933404324>

[23] URBANOWICZ, R. J., MOORE, J. H. *Learning Classifier Systems: A Complete Introduction, Review, and Roadmap.* Journal of Artificial Evolution and Applications [online]. 2009, 1-25 [cit. 13.12.2016]. DOI: 10.1155/2009/736398. ISSN 1687-6229. Dostupné z URL: <http://www.hindawi.com/archive/2009/736398/>

[24] CORTES, C., VAPNIK, V. *Support-vector networks.* 1995, Machine Learning. 20 (3): 273–297. [cit. 26.4.2017] DOI:10.1007/BF00994018

[25] SCHÖLKOPF, B., SMOLA, A.J., WILLIAMSON, R.C., BARTLETT, P.L. *New support vector algorithms.* 2000, Neural computation, 12(5), pp.1207-1245 [cit. 13.12.2016]. PMID: 10905814

[26] GANDHI, Y., PANIGRAHI, B. K., GUPTA, T. *Classification of post contrast T1 weighted MRI brain images using support vector machine* 2016, 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 2560-2563, ISBN 978-9-3805-4421-2

[27] HALDER, A., DOBE. O. *Detection of tumor in brain MRI using fuzzy feature selection and support vector machine* 2016, International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, pp. 1919-1923. doi: 10.1109/ICACCI.2016.7732331

[28] SUMITHRA, M. G., DEEPA, B. *Performance analysis of various segmentation techniques for detection of brain abnormality* 2016, IEEE Region 10 Conference (TENCON), Singapore, pp. 2056-2061. doi: 10.1109/TENCON.2016.7848387

[29] LIAO, Y., RUBINSTEYN, A., POWER, R., LI, J. *Learning random forests on the GPU* 2013, New York University, Department of Computer Science

[30] CHALUPA, D., *Supervised segmentation toolbox* [online] 2017 Dostupné z URL: <`https://github.com/chalupaDaniel/slicerSupervisedSegmentation`>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

BRATS	Brain Tumor Segmentation
CTK	Common Toolkit
DCMTK	DICOM Toolkit
DICOM	The Digital Imaging and Communications in Medicine
FLAIR	Fluid-attenuated Inversion Recovery
GUI	Grafické uživatelské rozhraní
ITK	Insight Toolkit
MRML	Medical Reality Modeling Language
SSL	Secure Sockets Layer
SVM	Support Vector Machines
VTK	Visualisation Toolkit
XML	Extensible Markup Language

SEZNAM PŘÍLOH

A Nastavení vývojového prostředí pro rozšíření 3D Slicer **57**

B Nastavení vývojového prostředí pro rozšíření Supervised Segmentation Toolbox **58**

C Přidávání algoritmů pro předzpracování a klasifikaci **59**

D Obsah přiloženého CD **60**

A NASTAVENÍ VÝVOJOVÉHO PROSTŘEDÍ PRO ROZŠÍŘENÍ 3D SLICER

Následující návod popisuje vytvoření a nastavení vývojového prostředí pro vytváření rozšíření 3D Sliceru ve verzi 4.5.0-1 tak, jak je standardní během psaní práce. Návod byl testován na operačním systému Windows 10 s použitím CMake verze 3.6.1, SlikSVN 1.9.4, Git 2.9.3.windows2, Qt 4.8.7 s podporou SSL, Visual Studio 13 Update 5. Byla testována kompilace Qt, 3D Slicer i rozšíření v Debug i Release módu, v 32- i 64-bitové verzi.

Prvním krokem je stažení potřebných programů. CMake je systém pro správu kompilačního procesu nezávislý na operačním systému a kompilátoru. V případě vývoje na Windows s Visual Studiem je použit pro vytvoření Solution a Project souborů. Návod na [17] doporučuje verzi vyšší, než 3.3.1. Dalším potřebným softwarem je stažení Git ve verzi vyšší, než 1.7.10. SlikSVN je poslední ze softwaru, který je potřeba ke konfiguraci Sliceru. Visual studio lze použít community version 2013, který je zdarma pro jednotlivce pro omezení.

Dalším bodem je kompilace Qt 4.8.7 s podporou Secure Sockets Layer (dále SSL). Binární soubory této verze Qt, které lze stáhnout na stránkách frameworku bohužel podporu SSL neobsahují. Naštěstí lze tento proces zautomatizovat příkazy z [21], které se zadávají do Powershell Visual Studia. Při kompilaci pro 64-bit verzi Qt pro použití v téže verzi Sliceru je třeba nastavit Powershell, aby používal 64-bit sadu nástrojů. Toho se docílí pomocí vcvarsall.bat.

Výpis A.1: Nastavení powershellu pro použití 64-bit sadu nástrojů

```
C:\> cd "Program Files (x86)\Microsoft Visual Studio 12.0\VC"
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC> vcvarsall amd64
```

V případě instalace Visual Studia do jiného, než standardního adresáře je kód třeba upravit. Pro kompilaci Slicer bude použit systém CMake. Nastaví se adresář, ve kterém jsou staženy zdrojové kódy a adresář, do kterého se vygenerují soubory Visual Studia. Konfigurace 3D Slicer vyžaduje pouze cestu ke spustitelnému souboru git a cestu ke Qt. V případě vytváření debug verze Slicer je při konfiguraci nutné povolit SimpleITK.

Vytváření rozšíření 3D Slicer je pomocí modulu Extension Wizard přímo z uživatelského rozhraní. Poté je opět potřeba vygenerovat projekt Visual Studia pomocí CMake.

B NASTAVENÍ VÝVOJOVÉHO PROSTŘEDÍ PRO ROZŠÍŘENÍ SUPERVISED SEGMENTATION TOOLBOX

Ve zkratce lze postup pro nastavení vývojového prostředí popsat jako:

1. Stažení dlib
2. Stažení knihovny boost s binárními soubory pro vybraný kompilátor
3. Stažení a kompilace knihovny shark
4. Doplnění konfiguračních záznamů modulu
5. Úprava zdrojových souborů boost
6. Kompilace rozšíření

Předpokladem pro začátek rozšíření modulu je zkompilovaný 3D Slicer v požadované verzi se všemi doprovodnými nástroji. Dalšími předpoklady jsou stažené knihovny dlib a shark, kde shark je ještě nutno zkompilovat. Kořenový adresář dlib stačí propojit s modulem pomocí konfiguračního záznamu nástroje CMake. Konfigurační záznam má název `DLIB_INCLUDE`. Minimální podporovaná verze je 19.3.

Ke kompilaci knihovny shark je potřeba sada knihovent boost v binární podobě, minimálně ve verzi 1.59, doporučeně 1.63. Po zkompilování knihovny shark opět pomocí sady nástrojů CMake a vybraného kompilátoru je třeba nastavit konfigurační záznam `SHARK_DIR` do kořenového adresáře shark. Posledním záznamem, který je třeba vyplnit je `BOOST_LIB`, který by měl odkazovat na adresář, v kterém se nachází binární soubory sady knihoven boost pro vybraný kompilátor.

V tuto chvíli by mělo být možné vygenerovat pomocí CMake soubory pro Visual Studio. Autor připomíná, že postup byl testovaný pouze na platformě windows pomocí integrovaného vývojového prostředí Visual Studio 2015, tedy jiné Visual Studio, než je potřeba ke kompilaci 3D Slicer. Ke zdárné kompilaci je ještě potřeba v době psaní práce vyřešit nekompatibilitu některých částí knihoven boost s předprocesorem Qt Meta-object Compiler. Toho se docílí ochranou, ukázanou ve výpisu B.1, upravující soubory has_binary_operator.hpp a has_prefix_operator.hpp v adresáři boost\type_traits_detail.

Výpis B.1: Ochrana pro vyřešení kompatibility boost a Meta-object Compiler

```
1 #ifndef Q_MOC_RUN // Na prvnim radku souboru
2 // ...
3 // Cely obsah souboru
4 // ...
5 #endif // Na poslednim radku souboru
```

C PŘIDÁVÁNÍ ALGORITMŮ PRO PŘEDZPRACOVÁNÍ A KLASIFIKACI

K rozšíření funkcionality je předpokladem zkompilovaný 3D Slicer a úspěšné zkompilování Supervised Segmentation Toolbox. Vytvoření vlastního předzpracovacího algoritmu programátor docílí děděním ze třídy `PreprocessingAlgorithm`, který se nachází v repozitáři ve složce *Preprocessing*. V této složce by taky měly být umístěny nové soubory. Pro úspěšné dědění třídy je potřeba vytvořit funkce `preprocess`, `serialize` a `deserialize`. První ze jmenovaných dostává parametry vstupní obraz, výstupní obraz a referenční číslo, které se Qt signálem `preprocessingDone` odešle po předzpracování obrazu. Předzpracovací algoritmus může mít také GUI. Zde je možné se inspirovat již vytvořeným mediánovým filtrem. Po vytvoření souborů nového předzpracovacího algoritmu je potřeba tyto soubory vložit do *SegmentationToolboxModule\Widgets\CMakeLists.txt*. Posledním krokem je vložení algoritmu do seznamu předzpracovacích algoritmů v konstruktoru třídy `PreprocessingAlgorithmList`, dle výpisu C.1

Výpis C.1: Přidání vlastního algoritmu do seznamu PreprocessingAlgorithmList

```
1 algorithms.append(QSharedPointer<NovyAlgoritmus>(new NovyAlgoritmus()));
```

Velice podobně lze přidávat i klasifikátory. Zde se dědí z třídy `Supervised Classifier` ve složce *Classifying*. Kromě obvyklých algoritmů serializace a deserializace je třeba přetížit metody `addVolumesToTraining` a `addVolumesToClassifying` a případně volitelně `startTraining`. U první jmenované je parametrem vektor vstupních MRML uzlů, kde každý uzel odpovídá obrazu, který je považován za jednu dimenzi učebních vektorů. Druhým parametrem této funkce je obraz obsahující pravdivostní data. Parametry funkce `addVolumesToClassifying` je opět vektor vstupních obrazů a druhým parametrem je výstupní obraz, kam se uloží značky klasifikovaných voxelů. Volitelně přetížitelnou metodu `startTraining` volá `VolumeManager` ve chvíli, kdy jsou všechna potřebná data předzpracována, načtena a odeslána klasifikátoru. Nový klasifikátor musí na konci učení zavolat Qt signál `trainingFinished` k odblokování uživatelského rozhraní. Na rozdíl od předzpracovacích algoritmů mají klasifikátory dvě GUI, kde jedno slouží k nastavení učení a druhé ke klasifikaci. Opět je třeba zdrojové soubory přidat do *SegmentationToolboxModule\Widgets\CMakeLists.txt* a do konstruktoru `ClassifierList` přidat třídu klasifikátoru dle výpisu C.2.

Výpis C.2: Přidání vlastního klasifikátoru do seznamu ClassifierList

```
1 classifiers.append(QSharedPointer<NovyKlasifikator>(new NovyKlasifikator()));
```

D OBSAH PŘILOŽENÉHO CD

V přiloženém CD čtenář nalezne binární soubory pro použití s 3D Slicer verze 4.6 na 64-bitovém operačním systému Windows, vypočtená data ve formátu csv, ukázky stc souborů, tento text a zdrojové kódy jak k rozšíření, tak ke zpracování přiložených dat v jazyce matlab (R2016a). Adresářová struktura je k nahlédnutí níže.

```
/ ..................................... kořenový adresář přiloženého CD
├── Bin ............................... binární soubory rozšíření
├── Data .............................. vypočtená data
│   ├── C SVM - dlib
│   ├── N SVM - dlib
│   └── Nahodny les - shark
├── Klasifikatory ..................... ukázky klasifikátorů
├── Text .............................. text práce ve formátu pdf
└── Zdrojove kody
    ├── SegmentationToolbox ........... kopie repozitáře k datu 11.5.2017
    └── Zpracovani dat ............... skripty a funkce ke zpracovani prilozenych dat
```