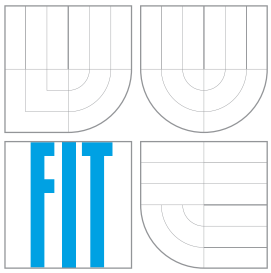
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ZPRACOVÁNÍ A VIZUALIZACE HMOTNOSTNÍCH SPEKTER

PROCESSING AND VISUALIZATION OF MASS SPECTRUMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ONDŘEJ BENEŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. TOMÁŠ MARTÍNEK, Ph.D.

BRNO 2014

Abstrakt

Jedna z nových technik v oblasti analytické chemie, které nalézá stále více uplatnění v praxi, je zobrazovací hmotnostní spektrometrie. Díky její schopnost zaznamenat zastoupení látek ve vzorku při analýze tkáně však vznikají až GB dat a tyto data je nutné zpracovávat programově. Cílem této práce je vytvořit aplikaci pro zpracování a vizualizaci dat z nového standardu imzML. Součástí práce je stručný úvod do problematiky hmotnostní spektrometrie, konkrétněji pak typu MALDI TOF a jsou zde popsány některé metody předzpracování hmotnostně spektrometrických dat. V práci je také nahlédnuto na současný stav existující software a na základě požadavků spolupracující laboratoře je navržena a implementována nová aplikace nejen umožňující zobrazení dat, ale i jejich předzpracování jako jsou například metody vyhlazování dat Savitzky-Golay, interní kalibrace či vyhledávání píků pomocí spojité vlnkové transformace. K závěru jsou i některé ukázky vizualizovaných dat z reálných experimentů.

Abstract

One of new techniques in the field of analytical chemistry, which has more and more practical use, is mass spectrometry imaging. With its ability to record representation of substances in samples during the tissue analyze arise problem with a lot of output data which needs to be handled programmatically. The goal of this work is to create an software for processing and visualization data of new standard imzML. As a part of the work, the field of mass spectrometry, primarily MALDI TOF mass spectrometry, is briefly introduced. There are also introduced some methods for mass spectrometry data preprocessing. The work also contains a summary of current state of available software for processing and visualization of mass spectrometry data. With requests from cooperating laboratory a novel software is designed and implemented, which besides the visualization itself, can preprocess the data for example data smoothing with Savitzky-Golay method, internal calibration or peak detection with continuous wavelet transformation. The software was successfully tested on real data sets.

Klíčová slova

hmotnostní spektrometrie, zobrazovací hmotnostní spektrometrie, vizualizace, imzML, předzpracování

Keywords

mass spectrometry, mass spectrometry imaging, visualization, imzML, preprocessing

Citace

Ondřej Beneš: Zpracování a vizualizace hmotnostních spekter, diplomová práce, Brno, FIT VUT v Brně, 2014

Zpracování a vizualizace hmotnostních spekter

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Tomáše Martiníka, Ph.D.

.....
Ondřej Beneš
27. května 2014

Poděkování

Rád bych na tomto místě poděkoval vedoucímu práce Ing. Tomáši Martiníkovy, Ph.D. za jeho čas, ochotu a cenné připomínky. Dále bych rád poděkoval prof. Janu Preislerovy a jeho spolupracovníkům z Masarykovy Univerzity za spolupráci a umožnění přístupu do Laboratoře bioanalytické instrumentace.

Obsah

Kapitola 1

Úvod

Hmotnostní spektrometrie s laserovou desorpcí/ionizací za účasti matrice ve spojení s průletovým analyzátorem je moderní bioanalytickou metodou využívanou pro určení hmotnosti biomolekul a k jejich identifikaci. Atraktivní aplikací této metody je její provedení ve zobrazovacím módu, který umožňuje poskytnout informaci o prostorovém zastoupení látek ve vzorku, nejběžněji v tkáňových řezech.

K získávání dat je využíváno přístrojů nazývaných hmotnostní spektrometry, jejichž vývoj sleduje moderní trendy v elektronice a informatice. Vysoká kvalita a v případě zobrazovací hmotnostní spektrometrie i velké množství spekter s sebou nese i velký objem dat (v řádu GB), který není v lidských silách manuálně zpracovávat a analyzovat v reálném čase.

Cílem této práce je prozkoumat možnosti využití informačních technologií a vytvořit software pro základní analýzu hmotnostních spekter a hyperspektrálních obrazců pro nekomerční hmotnostní spektrometr sestavený v Laboratořích bioanalytické instrumentace na Ústavu chemie Přírodovědecké fakulty Masarykovy univerzity. Tento software pomůže vědcům při další práci s daty.

První kapitola práce 2 vysvětluje principy a aplikace hmotnostní spektrometrie, především pak hmotnostní spektrometrie s laserovou desorpcí/ionizací za účasti matrice ve spojení s průletovým analyzátorem. Pozornost je věnována i zobrazovací hmotnostní spektrometrii. Druhá kapitola práce 3 popisuje teorii zpracování dat, tzv. hmotnostních spekter. Přibližuje konkrétní problémy, které je nutné při práci s daty z hmotnostního spektrometru řešit, aby jejich následná analýza dosáhla kvalitních výsledků. V další kapitole 4 se krátce podíváme jaký je současný stav software pro práci s daty z hmotnostních spektrometrů. Následující kapitola 5 popisuje samotnou implementaci software pro čtení spekter, jejich zpracování a zobrazení v uživatelském rozhraní. Nejprve je v této části věnována pozornost již existujícím softwarovým řešení používaných pro zpracování dat z hmotnostních spektrometrů s laserovou desorpcí/ionizací za účasti matrice a průletového analyzátoru a následně je popsána konkrétní implementace. V předposlední kapitole 6 bude popsáno několik experimentů naimplementované čtečky s daty a ukázány příklady výstupu ze software s testovacími i reálnými daty. V závěrečné kapitole 7 jsou pak shrnuty výsledky práce.

Kapitola 2

Hmotnostní spektrometrie

2.1 Historie

První náznaky hmotnostní spektrometrie se objevují roku 1886, kdy Eugen Goldstein pozoroval elektrické výboje v plynu pod nízkým tlakem. Zaznamenal při tom záření, které probíhalo od anody skrze kanálky v katodě. O pár let později Wildhelm Wien pozoroval, že silné elektrické či magnetické pole odchyluje takovéto záření a sestrojil přístroj s paralelním magnetickým a elektrickým polem, které rozdělovalo částice záření podle poměru jejich hmotnosti ku náboji (m/z) . Za otce hmotnostní spektrometrie je ale považován až Joseph John Thompson, který r. 1913 upravil Wienovu instrumentaci snížením tlaku a popsal tak izotopy neonu 20^{Ne} a 22^{Ne}. [27]

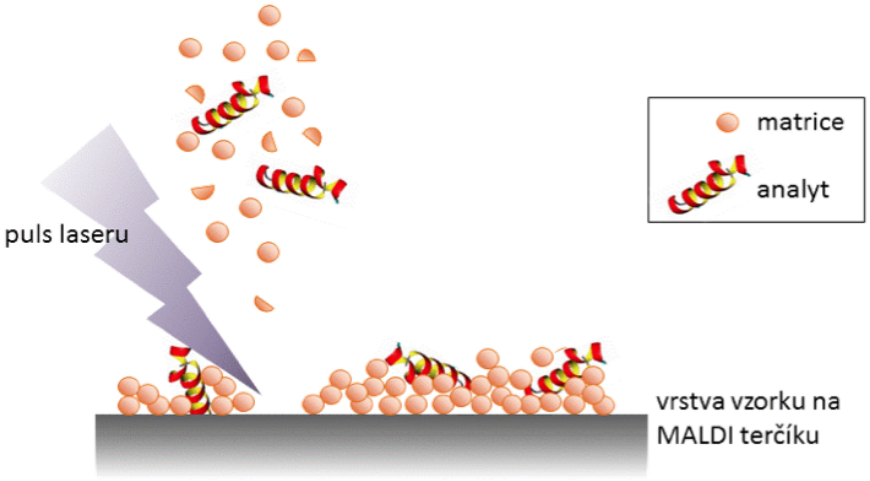
Během dvacátého století došlo k postupnému rozvoji hmotnostní spektrometrie, již ve 40. letech se hmotnostní spektrometry staly komerčně dostupné a získaly pevnou pozici esenciálních nástrojů fyziky i chemie. Zcela zásadní vliv na možnosti využití hmotnostní spektrometrie mělo ale objevení "měkkých" ionizačních technik: Koncem 80. let 20. století dvojice výzkumníků Franz Hillenkamp a Michael Karas a nezávisle na nich i Koichi Tanaka představili metodu laserovou desorpci/ionizaci za účasti matrice. Tato metoda ionizace, podobně jako ionizace elektrosprejem (z anglického electrospray ionization), kterou představil John Bennett Fenn, umožňuje analyzovat velké molekuly (např. proteiny, nukleové kyseliny) aniž by došlo k jejich destrukci či fragmentaci, která byla charakteristická pro tzv. tvrdé ionizačních metody. Hmotnostní spektrometrie využívající tyto ionizační techniky je dnes nezastupitelným nástrojem v oblasti proteomiky. V roce 2002 byla udělena Nobelova cena za rozvoj metod identifikace a strukturní analýzy biologických makromolekul. [11, 25, 8]

2.2 Princip a využití

Hmotnostní spektrometrie (z anglického mass spectrometry nebo také MS) je metodou analytické chemie. Základní princip metody spočívá v převedení zkoumané látky do plynného stavu a ionizaci na nabité částice, která je následována separací iontů podle jejich m/z v hmotnostním analyzátoru a jejich detekcí. Tímto lze získat zastoupení jednotlivých částic v měřeném vzorku a určit tak jeho složení. V současné době nachází uplatnění v odvětvích jako jsou chemie, fyzika, biologie, forenzní analýze, farmacie, medicíně či třeba i geologii.

2.3 MALDI

Laserová desorpce/ionizace za účasti matrice (MALDI; z anglického Matrix-Assisted Laser Desorption/Ionization) se využívá pro analýzu molekulové hmotnosti a identifikaci peptidů, proteinů, sacharidů, lipidů, nukleových kyselin či jiných makromolekul. Základní princip této ionizační techniky spočívá ve smíchání měřeného vzorku s tzv. MALDI matricí (malá organická látka, typicky kyselina). Na takovýto vzorek se pak působí v iontovém zdroji krátkými nanosekundovými laserovými pulsy. Procesy, které při tomto ději nastávají doposud nebyly zcela objasněny, ale předpokládá se, že molekuly matrice silně absorbují energii laseru, přičemž dochází k jejich desorpci (obrázek 2.1). Odpařující se částice matrice s sebou do plynné fáze strhávají molekuly vzorku. Současně dochází k ionizaci vzorku. Pro MALDI je typický vznik jednonásobně nabitých, tzv. pseudomolekulárních iontů $[M+H]^+$.



Obrázek 2.1: Schématická reprezentace procesu MALDI

2.4 Průletový analyzátor

Po ionizaci analytu v iontovém zdroji jsou ionty separovány v tzv. průletovém analyzátoru (TOF; z anglického Time-of-Flight). Ionty jsou nejdříve urychleny do letové trubice pomocí dvojice extrakčních mřížek, na které je přivedeno vysoké napětí. Potenciální energie urychlených iontů odpovídá kinetické energii iontů:

$$E_p = E_{kin} \rightarrow z \cdot U = \frac{m \cdot v^2}{2} \tag{2.1}$$

kde:

E_p ... potencionální energie,

E_{kin} ... kinetická energie,

U ... urychlovací napětí,

m ... hmotnost iontu,

z ... náboj iontu,

v ... rychlost.

Rychlost částic se nemění v driftové zóně bez silového pole a tak jsou ionty o různém m/z separovány na základě doby letu od iontového zdroje k detektoru. Pro stejně nabité částice platí, že lehčí částice dopadají na detektor dříve než částice těžší. Platí zde vztah:

$$m/z = \frac{2 \cdot e \cdot U \cdot t^2}{L^2} \tag{2.2}$$

kde:

m ... hmotnost iontu,

z ... náboj iontu,

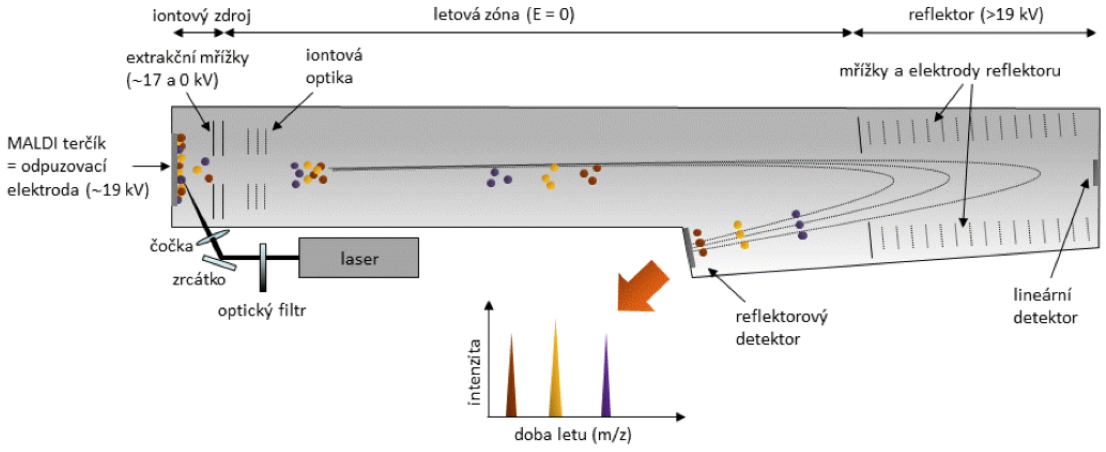
e ... elementární náboj,

U ... urychlovací napětí,

t ... doba letu iontu analyzátorem,

L ... délka takzvané driftové zóny.

Pro lepší rozlišení hmotností se používá několik různých metod, které korigují nestejnou distribuci počáteční kinetické energie vzniklou v důsledku ionizačního procesu. Jednou z nich použití iontového zrcadla na konci letové trubice neboli reflektoru. Ten je tvořen několika prstencovými elektrodami, na které je přivedeno napětí o stejné polaritě jako je napětí v urychlovací trubici avšak v opačné prostorové orientaci (obrázek 2.2). Tím pro částice o stejném m/z nastane, že částice s větší kinetickou energií do reflektoru proniknou hlouběji, zatímco ty s menší jej opustí dříve a všechny nakonec dopadnou na detektor současně. [21, 14]

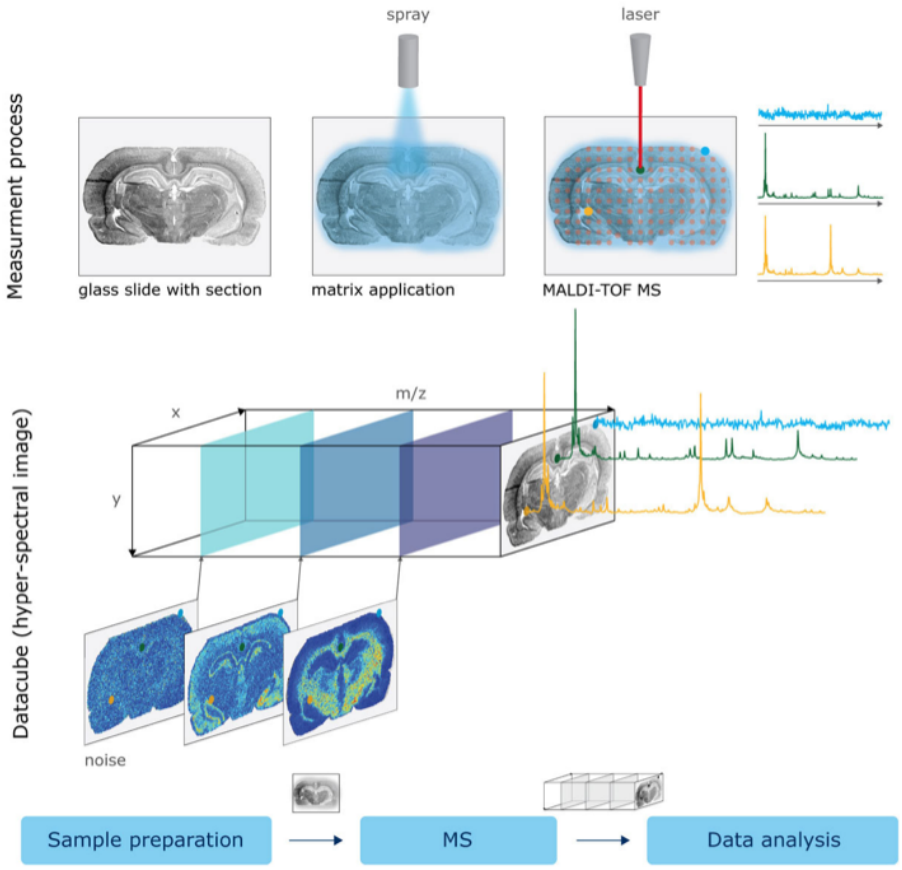


Obrázek 2.2: MALDI TOF s reflektorem

2.5 Zobrazovací hmotnostní spektrometrie

Zobrazovací hmotnostní spektrometrie (z anglické Mass Spectrometry Imaging neboli MSI) umožňuje rychlou detekci, lokalizaci a identifikaci molekul ve složitějších biologických vzorcích jako jsou třeba sekce zvířecích těl. MSI vzniklo potřebou získat prostorovou informaci o biologických molekulách, které lze získat běžnou hmotnostní spektrometrií.

Pro MSI jsou vhodné různé biologické vzorky počínaje od jednotlivých buněk, bakteriálních kolonií, vzorků biopsie, částí rostlin, zvířecí embrya, zvířecí orgány až po celé části zvířecích těl. Takový vzorek je nejprve zmrazen velmi nízkou teplotou a poté nařezán na tenké řezy. Na tyto vzorky je nanesena matrice a položené na terčíku jsou vkládány do hmotnostního spektrometru. Ten funguje stejným způsobem jako normální hmotnostní spektrometr, jen s tím rozdílem, že jednotlivé vzorky nejsou měřeny separátně, ale při akvizici dat je každému bodu (spektru) přiřazena jeho pozice, která po nasnímání celé oblasti zájmu následně umožní rekonstrukci tzv. hyperspektrálního obrazce (obrázek 2.3). [1]



Obrázek 2.3: Proces získání dat z tkáně pomocí MSI [1]

2.6 Drobeček

Hmotnostní spektrometr MALDI TOF, „Drobeček“ je nekomerčním, vlastními prostředky sestrojeným vysoce výkonným přístrojem s axiální konstrukcí (obrázek 2.4). Tento hmotnostní spektrometr byl původně prof. Preislerem zkonstruován na Northeastern University

v Bostonu. V roce 2009 byl věnován Masarykově univerzitě, zde znovu sestaven a vylepšen. Přístroj využívá pulzní, diodami pumpovaný Nd:YAG laser s násobičem frekvence o vlnové délce 355-nm a frekvenci do 4 kHz, motorizovaný přesun MALDI terčíku s rychlostí 5 mm/s, a vysokorychlostní 2 GS/s AD převodník pro záznam dat.

Jedinečným je přístroj také díky rychlému optickému skenovacímu systému, který umožňuje pohybovat desorpčním laserovým paprskem v časech pod 1 milisekundu. Zmíněné charakteristiky předurčují přístroj k velmi rychlým analýzám, zvláště pak pro MS imaging. Přístroj je schopen provádět zobrazovací hmotnostní spektrometrii téměř o řád rychleji, než současné běžně dostupné komerční přístroje. Mapu skládající se z 10 000 pixelů dokáže instrument nahrát za 11 minut. [3]

Obrázek 2.4: Přístroj v jeho současné podobě v laboratoři na Masarykově Univerzitě

Kapitola 3

Zpracování dat z hmotnostních spektrometrů

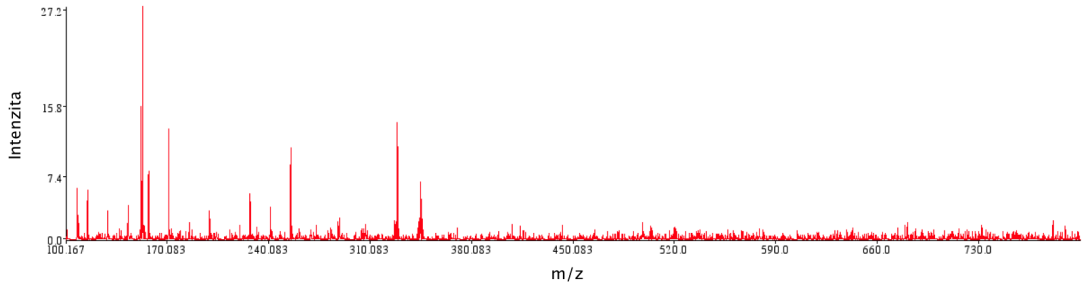
3.1 Výstup z hmotnostního spektrometru

Výstupem hmotnostního spektrometru typu MALDI-TOF je tzv. hmotnostní spektrum, tedy závislost signálu intenzity iontů dopadajících na detektor na čase, jak je popsáno v kapitole 2.4. V důsledku potřeby velmi přesných měření hmot bývají dnes výstupní data měření velmi obsáhlá. Jejich velikost se pohybuje od několika MB pro jednotlivá spektra až do řádů GB pro obrazce vytvořené pomocí zobrazovací hmotnostní spektrometrie. Velikost takového spektra je závislá na typu přístroje, přesnosti měření a velikosti měřeného vzorku.

3.2 Spektrum

Na spektrum se z pohledu datového lze dívat jako na tabulku. Taková tabulka je pak složena z časové osy vyjadřující čas, za který daný iont dorazil na detektor. Tato doba může být při znalosti geometrie hmotnostního spektrometru přepočtena na přibližnou hodnotu m/z, která bývá označována v jednotkách Dalton (Da). Druhý rozměr je intenzita signálu iontů, které v daný čas dorazily na detektor. Data se běžně vynáší do grafu (obrázek 3.1), kde osa x označuje rozměr m/z a osa y rozměr intenzity signálu.

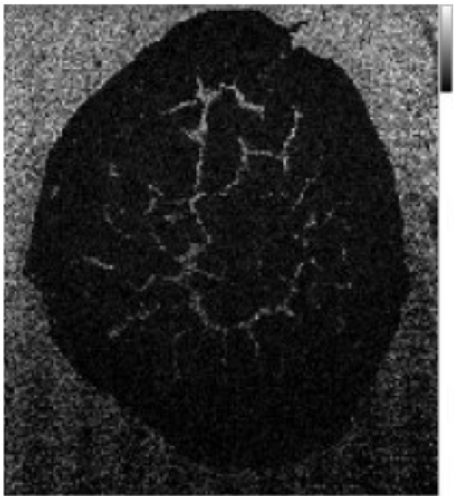
V závislosti na typu přístroje, vzorkovací frekvenci může mít takovéto spektrum v m/z ose desítky až stovky tisíc bodů.



Obrázek 3.1: Hmotnostní spektrum vynesené do grafu

3.3 Hyperspektrální obrázek

V případě MSI lze na výstup měření pohlížet jako na obrázek. Měřený vzorek je bod po bodu nasnímán v předem definované oblasti. Každý tento bod je pak reprezentován jedním spektrem. Tyto spektra mohou mít stejnou časovou osu, ale mohou mít také časovou osu odlišnou. Nejjednodušeji a nejnázorněji se data zobrazují do podoby obrázku. Nejprve je však nutné si zvolit hodnotu m/z, pro kterou bude daný obrázek vytvořen. Každý pixel takového obrázku pak reprezentuje intenzitu signálu pro dané m/z v souřadnicích x, y. Pokud takové takové intenzity vyneseme do 2D mapy, lze vytvořit obrázek 3.2.



Obrázek 3.2: Hyperspektrální obrázek řezu močového měchýře [20]

Jelikož hmotnostní spektrometr pracuje na úrovni molekul, bývá výstup takového měření velmi obsáhlý co se týče počtu dat. Pokud se měří například vzorek o velikosti 100 krát 100 bodů a každé měření jednoho bodu má například sto tisíc bodů v ose m/z, pak je počet hodnot v takovém obrázku roven 1 miliardě. Měření se většinou zaznamenávají čísly s plovoucí čárkou. Pokud se použije dvojí přesnost tak čistě velikost takovýchto hrubých dat jednoho takového měření je kolem 7630MB, obvykle to však bývá o něco víc, vzhledem k některým metadatům, které jsou pro následnou analýzu nezbytné.

3.4 Formát dat

mzML

Rozšířením využití hmotnostní spektrometrie vznikl požadavek sdílet naměřená data mezi jednotlivými pracovními skupinami. Problém je, že s komerčními hmotnostními spektrometry je dodáván software vytvářený danou společností, který pracuje s proprietárním formátem. V roce 2008 pracovní skupina HUPO PSI vyvinula společný standard pro ukládání dat z hmotnostních spektrometrů. Byl nazván mzML a byl určen pro uložení jednoho spektra s metadaty. V roce 2009 byla pak vydána nová, aktuální verze standardu verze 1.1.0. [15]

Tento formát dat je postaven na technologii XML. Hmotnostní spektrometrie má mnoho typů a každé měření lze provádět různým způsobem, je tedy zapotřebí ukládat strukturovaně spousty metadat (obrázek 3.3). Tyto metadata popisují samotný experiment, prostředí

Obrázek 3.3: Schématické zobrazení struktury mzML souboru s klíčovými elementy [15]

či podmínky v jakém experiment vznikal či jiné. Jednotlivé položky metadat nejsou v tomto souboru definovány a místo toho je u nich uveden pouze odkaz do kontrolovaného slovníku, kterým je takzvaný OBO soubor (obrázek 3.4). Kromě metadat jsou v souboru uloženy také samotné naměřené data v binární podobě zakódována pomocí *base64*. Pro rychlejší přístup k datům jsou nejdřív v souboru uloženy data o pozici binárních dat a až na samotném konci jsou umístěny binární data, aby nebylo nutné načítat všechny data, ale bylo možné přistoupit ke konkrétním částem.

OBO

Formát OBO je textový formát reprezentující ontologický jazyk. Koncept, který tyto soubory reprezentují modeluje podmnožinu konceptů v deskriptivní logickém jazyce OWL s některýma rozšířeními pro modelování metadat a jiných konceptů, jenž nejsou podporovány v jiných jazycích. Samotný formát je možné číst v čisté textové formě a nebo pomocí programu OBO-Edit, který ontologie zobrazí ve formě stromových struktur. Ukázka jednoho záznamu z OBO souboru:

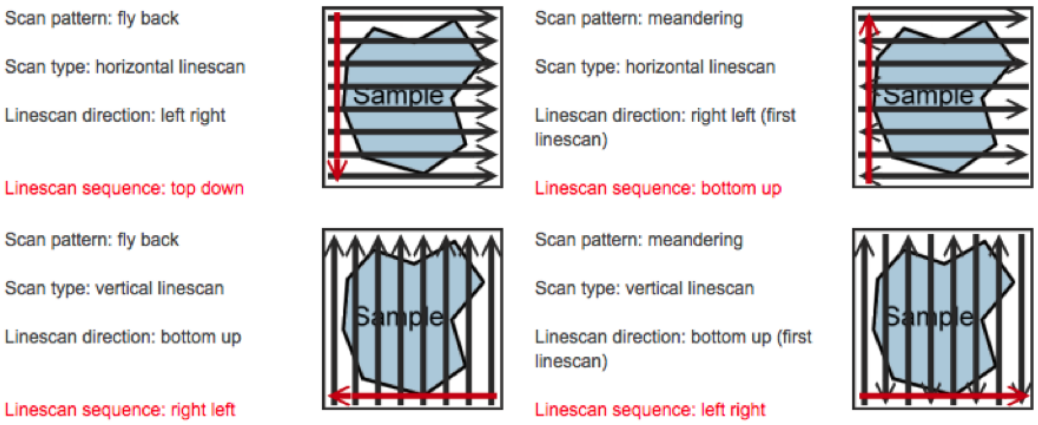
```
[Term]
id: IMS:1000003
name: ibd binary type
def: 'Describes type of the binary (ibd) file .' [COMPUTIS:IMS]
```

```
relationship: part_of IMS:0000000 ! Imaging Mass Spectrometry Ontology
```

imzML

Formát mzML je však pro potřeby zobrazovací hmotnostní spektrometrie nepraktický a tak v rámci evropského projektu COMPUTIS v roce 2012 za tímto účelem vznikl standard pro ukládání dat ze zobrazovací hmotnostní spektrometrie s názvem imzML. Tento formát přímo vychází z formátu mzML. Data zobrazovací hmotnostní spektrometrie jsou v tomto standardu oproti původnímu mzML souboru uloženy ne do jednoho, ale do dvou souborů. Jeden soubor obsahuje metadata ve formátu XML a druhý obsahuje samotné binární data z měření.

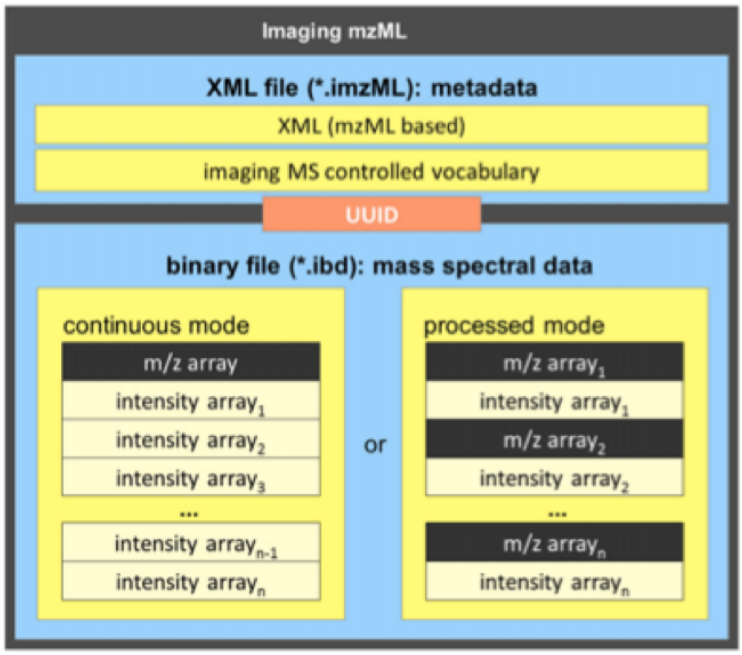
XML soubor s koncovkou imzml obsahuje všechny příslušné informace týkající se konkrétního experimentu. Formát tohoto souboru zůstal téměř stejný jako formát souboru mzML, také jsou zde použity odkazy do kontrolovaného slovníku. Jsou zde ale i rozdíly a to hlavně z důvodů různé interpretace výsledných dat. Byly zde doplněny metada například o x/y pozici, konfigurace skenování (obrázek 3.4) či velikost jednotlivých pixelů. Namísto samotných dat jsou zde také uloženy ofsety k jednotlivým spektrům pro rychlejší přístup.



Obrázek 3.4: Možné varianty konfigurace skenování [20]

Binární data mají koncovku ibd a obsahují samotné naměřené hodnoty. Z důvodu efektivního ukládání dat můžou tyto data být uloženy ve dvou různých módech: nepřetržitý (continuous) a zpracovaný (processed). Nepřetržitý mód předpokládá pro každé spektrum, nebo také pro každý bod měření stejné hodnoty m/z a tyto hodnoty jsou v binárních datech uloženy pouze jednou na začátku (obrázek 3.5). Tento mód používá většina MALDI-TOF hmotnostních spektrometrů. Díky takovému uložení dojde k velké úspoře dat. Oproti tomu druhý mód ukládá pro každé spektrum hodnoty m/z zvlášť. Výhodou tohoto módu je větší přesnost a věrohodnost dat, ale oproti nepřetržitému módu má výsledný soubor téměř dvojnásobnou velikost.

Oba tyto soubory lze spolu jednoznačně identifikovat pomocí jedinečného univerzálního identifikátoru (UUID), který je obsažen jak v metadatech, tak v binárních datech. Skládá se z 16 bajtů a v binárních datech je uložen hned na začátku. Dle RFC 4122 má tento identifikátor bajty uloženy v pořadí big-endian oproti tomu zbytek binárního souboru je vždy uložen s endianitou little-endian.



Obrázek 3.5: Struktura souborů standardu imzML [20]

3.5 Předzpracování dat

Jedno měření v rámci MALDI-TOF může obsahovat desítky, stovky a až tisíce hmotnostních spekter. Každé takové spektrum obsahuje tisíce intenzit reprezentující předem neznámý počet analytů ve formě píků. Jakýkoliv pokus dát těmto datům smysl vyžaduje nejprve nízko úrovňové předzpracování k zajištění přesnějšího získání pozic a velikostí jednotlivých píků. V opačném případě může dojít ke značnému zkreslení a ke zhoršení výsledné analýzy vzorku. Předzpracování hmotnostních spekter se skládá z několika úkolů, typicky:

- kalibrace,
- filtrování k odstranění šumu,
- korekce základny,
- normalizace,
- detekce píků.

Kalibrace

Existují dva základní typy kalibrací: externí a interní. Při externí kalibraci se používá standardní vzorek, který se umístí na jedno či více míst a parametry kalibrace jsou získány z tohoto standardního vzorku. Tyto parametry jsou pak následně použity ke kalibraci dalších vzorků v jiných místech. Oproti tomu interní kalibrace se provádí pomocí standardního vzorku smíchaného společně se vzorkem. Hodnoty píků standardních molekul jsou identifikovány a hmoty těchto molekul jsou použity ke kalibraci celého spektra. Externí kalibrace

je vhodná, ale často není dostatečně přesná. Oproti tomu interní kalibrace může být přesná, ale i ta má své nevýhody. Je nutná základní znalost o vzorku k určení, jaké množství standardních látek by mělo být přidáno do vzorku pro interní kalibraci. Pokud je totiž množství standardních látek nedostatečné, píky standardních látek ve spektru není možné identifikovat. Pokud je množství zase příliš velké, měřený vzorek může být potlačen. Může také nastat překryv píků standardních molekul a měřeného vzorku. Mohou se objevit také neočekávané či neznámé molekuly jako důsledek reakce mezi měřeným analytem a standardní látkou. [31]

Interní kalibrace

V měřené směsi se označí známé píky a určí se jejich reálná hodnota. Získáme tak množinu obsahující velikosti chyb v jednotlivých částech funkce spektra. Nejjednodušší způsob interní kalibrace je za pomocí lineární regrese metodou nejmenších čtverců. Uvažujme funkční závislost $f(x)$ kde x značí hodnotu m/z, kterou chceme přepočítat na novou kalibrovanou hodnotu:

$$f(x) = ax + b \tag{3.1}$$

Dosazením hodnot množiny chyb a množiny hodnot m/z do rovnic získáme parametry a a b:

$$a = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{3.2}$$

$$b = \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{3.3}$$

Pomocí takto získaných parametrů pak lze vypočítat lineárně kalibrované hodnoty [28]. Proložení bodů přímkou není vždy dostatečně přesná aproximace a další možný způsob výpočtu správných bodů je kalibrace pomocí polynomické regrese [30]. Princip je podobný jako lineární regrese, ale namísto přímky kalibrační body proložíme křivkou. Výpočet parametrů polynomu lze pomocí matice

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \begin{bmatrix} \sum x_i^{2k} & \cdots & \sum x_i^{k+1} & \sum x_i^k \\ \vdots & \ddots & \vdots & \vdots \\ \sum x_i^{k+1} & \cdots & \sum x_i^2 & \sum x_i \\ \sum x_i^k & \cdots & \sum x_i & n \end{bmatrix} \begin{bmatrix} p_k \\ \vdots \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} \sum y_i x_i^k \\ \vdots \\ \sum y_i x_i \\ \sum y_i \end{bmatrix} = \mathbf{A}^T\mathbf{b}. \tag{3.4}$$

Vezmeme-li kupříkladu polynom stupně dva, tedy kvadratickou funkci

$$f(x) = ax^2 + bx + c \tag{3.5}$$

bude matice pro výpočet parametrů vypadat následovně

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \begin{bmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 \\ \sum x_i^3 & \sum x_i^2 & \sum x_i \\ \sum x_i^2 & \sum x_i & n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum y_i x_i^2 \\ \sum y_i x_i \\ \sum y_i \end{bmatrix} = \mathbf{A}^T\mathbf{b}, \tag{3.6}$$

Normalizace

Normalizace je proces násobení hmotnostních spekter s faktorem škály intenzit k zvětšení či zmenšení intervalu osy intenzit. Je používána k projekci spekter různých intenzit na jednu společnou škálu intenzit. V zobrazovací hmotnostní spektrometrii je normalizace používána k systematickému odstranění artefaktů, které ovlivňují intenzitu. Ve všech analýzách hmotnostní spektrometrie složitějších vzorků může docházet k potlačení iontů. Jedna molekula může potlačit jinou. Pokud toto nastane, pozorované intenzity neodpovídají reálné koncentraci ve vzorku. Neaplikováním normalizace může vést k špatné výsledné analýze. Normalizace však musí být aplikovaná se znalostí, že rozdíly biologických vzorků jsou systémové chyby. [6]

Jedna z hlavních metod normalizace hmotnostních spekter je pomocí celkového proudu iontů neboli TIC (z anglického Total Ion Current). V této metodě jsou všechny spektra poděleny jejich TIC tak, že všechny spektra mají stejný integrál. Metoda je založena na předpokladu, že čísla ve spektrech jsou porovnatelná. Hmotnostní spektrum si lze představit jako vektor hodnot intenzit:

$$s = x_1, x_2, \dots x_n \tag{3.7}$$

Normalizace pak tento vektor dělí normalizačním faktorem f:

$$s_{norm} = \frac{1}{f} s \tag{3.8}$$

Normalizace TIC je speciálním případem p-normy :

$$f = (\sum_i |x_i|^p)^{\frac{1}{p}} \tag{3.9}$$

Pro $p = 1$ je normalizace založena na sumě všech intenzit v hmotnostním spektru. Se zvyšujícím P mají vyšší intenzity větší vliv na výsledek normalizace. Tento efekt lze pozorovat pro hodně zašuměná spektra. Při maximální normě jsou nejvyšší intenzity v zašuměném spektru normalizovány na stejnou úroveň jako je nejvyšší intenzita vrcholku ostatních spekter.

Protože výpočet úrovně šumu může ovlivnit operace jako vyhlazování, které jsou často součástí předzpracování hmotnostních spekter, používá se také normalizace založená na mediánu, která je robustnější pro takovéto metody předzpracování. Medián všech intenzit ve spektru je použit k výpočtu normalizační konstanty

$$f = median(x_i) \tag{3.10}$$

Vyhlazování

Hmotnostní spektra jsou díky elektromagnetickým a chemickým vlivům ovlivněna šumem. Před zpracováním takovýchto spekter je nutné tento šum nejprve odstranit. K tomu se běžně používají dolnopropustní filtry.

Klouzavý průměr

Nejjednodušší technikou pro vyhlazování signálů se stejně vzdálenými body je klouzavý průměr. Pole hodnot $x_1, x_2, \dots, x_n$ lze převést na pole vyhlazených dat. Vyhlazené body h_k jsou průměrem liché posloupnosti $2n + 1$ bodů původních hodnot
$x_{k-n}, x_{k-n+1}, \dots, x_{k-1}, x_k, x_{k+1}, \dots, x_{k+n-1}, x_{k+n}$

$$h_k = \sum_{i=-n}^{n} \frac{x_{k+i}}{2n+1} \tag{3.11}$$

Liché číslo lze jinak nazvat jako šířku filtru. Čím více je filtr širší, tím více vyhlazuje data.

Savitzky-Golay

Vyhlazovací filtr Savitzky-Golay byl v roce 1964 představen Abrahamem Savitzky a Marcel J. E. Golay [19]. Byl vytvořen neboť vyhlazování pomocí klouzavého průměru a jemu podobných technik nebyly dostatečně vhodné pro extrakci stanovených charakteristik píků spekter. Jednoduše řečeno, klouzavý průměr tyto píky roztahuje do šíři na základě velikosti definovaného okna.

Hlavní myšlenka spočívala ve vyhnutí se problémových vlastností vznikajících při použití klouzavého průměru jako jsou relativní maxima, minima a výše zmiňovaná šířka. Jako vhodné řešení se ukázalo aplikování lineární regrese polynomu jednotlivě pro každý vzorek a pak následné zhodnocení tohoto polynomu přesně v místech vzorku. Toto je klíčový bod v samotné metodě. Výpočet regrese pro každý bod zvlášť je časově náročné, lze však snížit pokud má regrese polynom konečného stupně. Poté totiž stačí koeficienty vypočítat pouze jednou pomocí konvoluce diskrétního vzorku vstupních dat s koeficienty vektoru.

K odvození algoritmu vyhlazování předpokládejme množinu bodů stejně od sebe vzdálených, které jsou lineární kombinací x_n bodů kde $n = 1, ..., M$

$$h_k = \sum_{j=-n_L}^{n_R} c_j x_{k+j} \tag{3.12}$$

kde:

n_L ... číslo označující vzorky nalevo,

n_R ... číslo označující vzorky napravo,

k ... index středu.

Lze si všimnou podobnost s klouzavým průměrem, kde jsou ale všechny koeficienty c_n stejné $c_n = 1/(n_L + n_R + 1)$. Cílem algoritmu je tyto koeficienty najít takové, které zachovají tvar a vlastnosti ve vzorku. Lze také říct, že chceme aproximovat základní funkci klouzavého průměru tak, aby nebyla použita konstanta, ale polynom vyššího stupně. Pro každý bod x_n vybereme vhodný polynom metodou nejmenších čtverců pro všechny body $n_L + n_R + 1$ v plovoucím okně a zvolíme h_k na hodnotu polynomu na pozici k. Díky tomu, že metoda nejmenších čtverců zahrnuje pouze inverzi matice, koeficienty jsou lineární kombinací samotných dat a tudíž je lze vypočítat dopředu. [10, 17]

K odvození těchto koeficientů předpokládejme že h_0 lze získat tak, že se snažíme vhodně najít polynom stupně M do k, konkrétně $a_0 + a_1 i + \cdots + a_M i^M$ pro hodnoty $x_{-n_L}, \ldots x_{n_R}$. Poté h_0 bude hodnotou polynomu $i = 0$, konkrétně a_0. Rovnice vektorů a_j ve vztahu k vektoru x_n v maticové formě pak je

$$(\mathbf{A}^T \mathbf{A}) \cdot a = \mathbf{A}^T \cdot f \tag{3.13}$$

nebo také

$$a = (\mathbf{A}^T\mathbf{A})^{-1} \cdot (\mathbf{A}^T \cdot f) \tag{3.14}$$

Konkrétní forma těchto matic pak je

$$\mathbf{A}^T\mathbf{A}_{ij} = \sum_{k=-n_L}^{n_R} \mathbf{A}_{ki} \cdot \mathbf{A}_{kj} = \sum_{k=-n_L}^{n_R} k^{i+j} \tag{3.15}$$

a

$$\mathbf{A}^T \cdot f = \sum_{k=-n_L}^{n_R} \mathbf{A}_{kj} \cdot f_k = \sum_{k=-n_L}^{n_R} k^j \cdot f_k \tag{3.16}$$

Koeficienty c_n jsou pak komponenty a_0 kde f je nahrazeno jednotkovým vektorem e_n kde $-n_L \leq n \leq n_R$:

$$c_n = (\mathbf{A}^T \cdot A)^{-1} \cdot (\mathbf{A}^T \cdot e_n) = \sum_{m=0}^{M} (\mathbf{A}^T \cdot \mathbf{A})^{-1}{}_{0_m} n^m \tag{3.17}$$

Vyhledávání píků

Jeden z dalších možných kroků předzpracování hmotnostních spekter proteomické analýzy je vyhledávání píků. Použití této techniky ovlivňuje následné procesy. Vzhledem ke složitosti signálu a vícero zdrojů šumu v hmotnostních spektrech je pravděpodobná vysoká míra falešně pozitivních nálezů, což může způsobit problémy v následné interpretaci dat. Samotné vyhledávání píků je prvním krokem před identifikací oblastí zájmu.

Vyhledávání pomocí vlnkové transformace

Obecně se u hmotnostních spekter provádí předzpracování vyhlazováním a odečtením základny, zvolíme-li však vhodný algoritmus, tyto kroky lze vynechat. Jedním z těchto vhodných přístupů může být algoritmus odpovídajících vzorů využívající spojitou vlnkovou transformaci. Tento algoritmus lze využít přímo na surová data bez předzpracování a výsledek bude více konzistentní pro různá spektra. Algoritmus si zachovává nízkou míru falešně pozitivních nálezů díky transformaci do vlnkového prostoru.

Píky v hmotnostních spektrech mají určité charakteristiky a vzory. Šířka a výška píků se může měnit díky překryvu s jinými píky či s šumem. Tyto problémy lze vyřešit použití vlnkové transformace. Namísto přímé detekce píků algoritmus nejprve identifikuje takzvané hřbety v 2D matici koeficientů vlnkové transformace a využije je k zjištění efektivnímu poměru šumu k signálu a k identifikace píků. Identifikací píků a přiřazení poměru šumu k signálu ve vlnkovém prostoru jsou problémy se základnou a šumem odstraněny. [7]

Spojitá vlnková transformace

Metody vlnkové transformace lze třídit do dvou kategorií a to diskrétní či spojitá. Diskrétní pracuje nad škálou a pozicemi založených na druhé mocnině. Samotná diskrétní vlnková transformace je hojně používaná v kompresi dat, protože je více efektivní a dostatečná pro přesnou rekonstrukci. Spojitá vlnková transformace umožňuje transformaci pro každou škálu a poskytuje tím dodatečné informace pro porovnávání vzorů. Navíc zde není potřeba ortogonální vlnka. Matematicky lze spojitou vlnkovou transformaci vyjádřit

$$C(a,b) = \int_R s(t)\Psi_{a,b}(t)dt, \Psi_{a,b}(t) = \frac{1}{\sqrt{a}}\Psi(\frac{t-b}{a}), a \in R^+ - 0, b \in R \tag{3.18}$$

kde:

$s(t)$ ... signál,

a ... škála,

b ... translace,

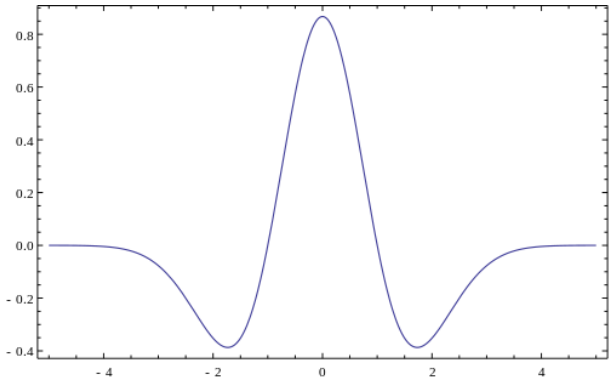
$\Psi(t)$ ... mateřská vlnka,

$\Psi_{a,b}(t)$ ... škálovaná a posunutá vlnka,

C ... 2D matice vlnkových koeficientů.

Koeficienty pak odráží porovnání vzorů mezi signálem s a $\Psi_{a,b}t$. Čím jsou koeficienty vyšší, tím dochází k lepšímu porovnání. Změnou škály a pak může $\Psi_{a,b}(t)$ odpovídat vzorům na různých škálách bez nutnosti vyvolávat složité nelineární prokládání křivkou. [5]

K vyhledávání píků je nutné sledovat efekt změny v šířce a výšce píků, které jsou škálovány a posunuty vlnkou $\Psi_{a,b}(t)$. Za účelem získání lepšího výkonu by vlnka měla mít základní vlastnosti píku jako je aproximace symetrie a jeden hlavní kladný pík. Jako vhodná se ukazuje takzvaná vlnka mexického klobouku (obrázek 3.6), která je druhou derivací hustoty Gaussova rozložení a je na intervalu $[-5, 5]$.



Obrázek 3.6: Vlnka nazývaná mexický klobouk [29]

Bez jakéhokoliv škálování ($a = 1$) vlnka poskytuje nejlepší shodu pro píky s šířkou velikosti zhruba dva. Pro vyšší škály a_1 pak shodu pro větší šířku píku $2a_1$. Pro spektra v hmotnostní spektrometrii mají koeficienty na každé škále lokální maxima kolem středu píku. Počínaje škálou $a = 1$ amplituda lokálního maxima roste spolu se škálou spojité vlnkové transformace a dosáhne maxima, když škála nejlépe odpovídá šířce píku. V 3D prostoru si toto lze představit jako hřbet, právě pokud vizualizujeme 2D matici koeficientů spojité vlnkové transformace s amplitudou oné transformace jako třetí dimenzi. Tímto se problém hledání píků transformuje na problém hledání hřbetů v 2D matici koeficientů a tím je méně náchylný na lokální minima a více robustní oproti změnám v koeficientech hledaného prostoru. Sledováním tohoto hřbetu ve vlnkovém prostoru lze tedy dohledat informace o domnělých pících.

Proces identifikace píků

Na obrázku 3.8 lze vidět 2D matici koeficientů vlnkové transformace. Čím více je barva více žlutá, tím větší je amplituda a naopak zelená značí menší amplitudy. Při pohledu na 3.7 lze vidět, že hřeby odpovídají s píky ve spektru (obrázek 3.8).

Hřbet může být identifkován spojením lokálních maxim koeficientů spojité vlnkové transformace na každé úrovni škály. Nejprve je lokální maximum identifikováno pro každou takovou úroveň. Dalším krokem pak je spojení těchto maxim do čar, což reprezentuje hřbety, které se snažíme nalézt 3.9.

Předpokládejme pak, že 2D matice koeficientů spojité vlnkové transformace je $N \times M$, kde N je počet škál a M je délka spektra. Proces identifikace je pak následující:

1. Inicializace čar hřbetu na základě lokálních maxim identifikovaných na nejvyšší škále, například řádek n, kde $n = N$ v matici koeficientů a počáteční mezeře hřbetu 0
2. Pro každou čáru hřbetu s mezerou menší než určitý práh hledáme nejbližší maximum sousedních škál. Pokud takový bod nenalezneme, zvětšíme mezeru, v opačném případě ji nastavíme na 0.
3. Uložíme čáry hřbetů, jenž mají počet mezer větší než určitý práh a odstraníme je z hledání
4. Maxima, které nejsou propojeny body z vrchních škál budou iniciovány jako nové hřbety.
5. Opakujeme krok 2-4 dokud nedosáhneme nejmenší škálu neboli řádek $n = 1$ v matici koeficientů

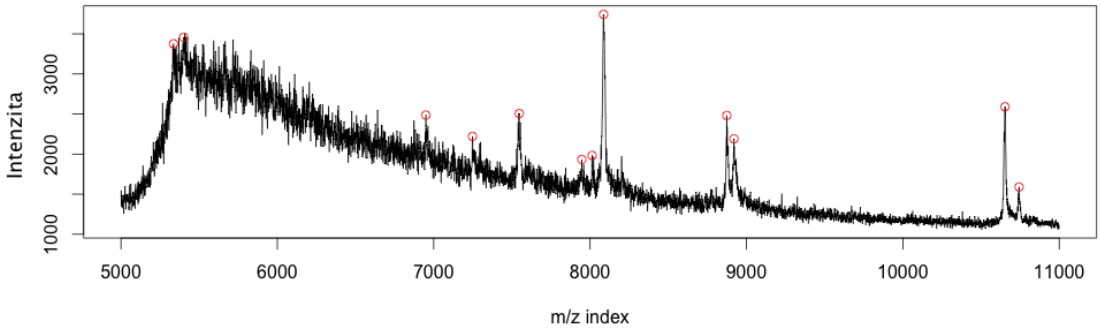
Takto identifikované hřbety jsou vidět na obrázku 3.9, kde barva značí relativní sílu jednotlivých koeficientů. Žlutá barva značí sílu blížící se nule a modrá naopak vyšší.

Před identifikací samotných píků je nutné nejprve definovat, co je vlastně poměr signálu k šumu. Na základě předpokladu, že spektrum má charakteristický tvar, je síla signálu píku definována maximem koeficientů vlnkové transformace na čáře hřbetu určitého rozsahu škál. Co se týče šumu, předpokládáme, že je složen z pozitivních nebo negativních píků, které jsou velmi úzké. Koeficienty v nízkých škálách jsou dobrým odhadem úrovně. Lokální úroveň šumu píku je definována jako 95% kvantil absolutní hodnoty koeficientu v rámci lokálního okna obklopující pík. Minimální úroveň šumu, může být poskytnuta, aby se zabránilo úrovni šumu blízké nule, což může nastat v oblastech kde je signál hladký. A tedy poměr signálu k šumu je definován jako poměr odhadu síly signálu píku a lokální úroveň šumu píku.

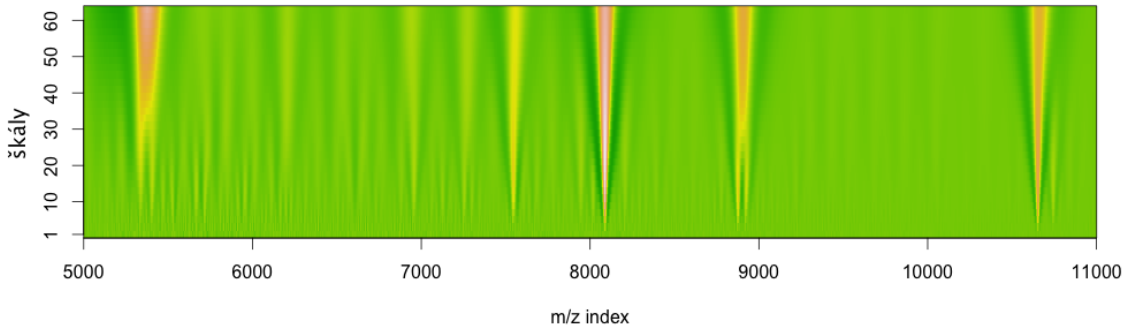
Pro identifikaci hlavních píků:

1. Škála odpovídající maximální amplitudě na čáře hřbetu je proporcionální k šířce píku a měla by být v určitém rozsahu
2. Poměr signálu k šumu by měl být větší než určitá úroveň
3. Délka čar hřbetu by měla být větší než určitá úroveň

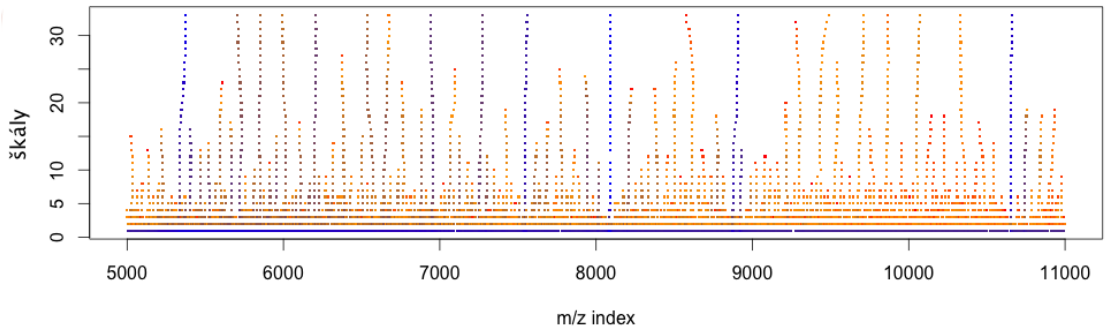
Často jsou okolo velkých píků malé píky, které jsou běžně označovány jako adukty polypeptidů s molekulami matrice nebo jejími fragmenty. Tyto píky mají kratší čáry hřbetu než hlavní silné píky. Redukováním úrovně z bodu 3 v prostoru obklopujícím hlavní píky, mohou být tyto malé píky identifikovány.



Obrázek 3.7: Identifikované vrcholky s poměrem šumu k signálu > 3



Obrázek 3.8: Koeficienty spojité vlnkové transformace



Obrázek 3.9: Identifikované čáry hřbetů založené na spojité vlnkové transformaci

Uvedený algoritmus poskytuje dvě možnosti odhadu pozice píku. Jedna vychází z čar hřbetu z nejvyšších škál k nejnižším. V tomto případě pozice na nižších škálách určuje maximum píku, druhý způsob je pak pomocí odhadu těžiště píku na základě koeficientů spojité vlnkové transformace v určitém rozsahu škál v čáře hřbetu. První metoda poskytuje

podobné výsledky jako běžné metody používané pro detekci píků. Druhá metoda je více konzistentní napříč různými spektry.

Kapitola 4

Současný stav

Jelikož je standard imzML 3.4 relativně mladý, dá se předpokládat, že doplnění jeho podpory do stávajících software nějaký čas zabere. V této kapitole bude zběžně prozkoumána aktuální stav podpory tohoto formátu. Počet software a různých pomocných knihoven pro hmotnostní spektrometrii je obecně poměrně velký. Proto se tato kapitola zaměří pouze na software, který se týká se zobrazovací hmotnostní spektrometrií a má případně podporu standardu imzML. Jsou zde prozkoumány aplikace Datacube Explorer z nizozemské instituce AMOLF, BioMap a MSImageView obě od švýcarské farmaceutické společnosti Novartis International AG, MALDIVision od společnosti PREMIER Biosoft, MSiReader z laboratoří W.M. Keck FT-ICR Mass Spectrometry Laboratory a komerční aplikaci flexImaging od firmy Bruker. Na konci se bude stručně přiblížena aplikace pro hmotnostní spektrometrii mMass.

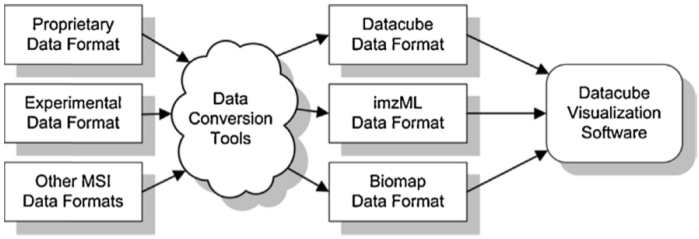
4.1 Aplikace

Datacube Explorer

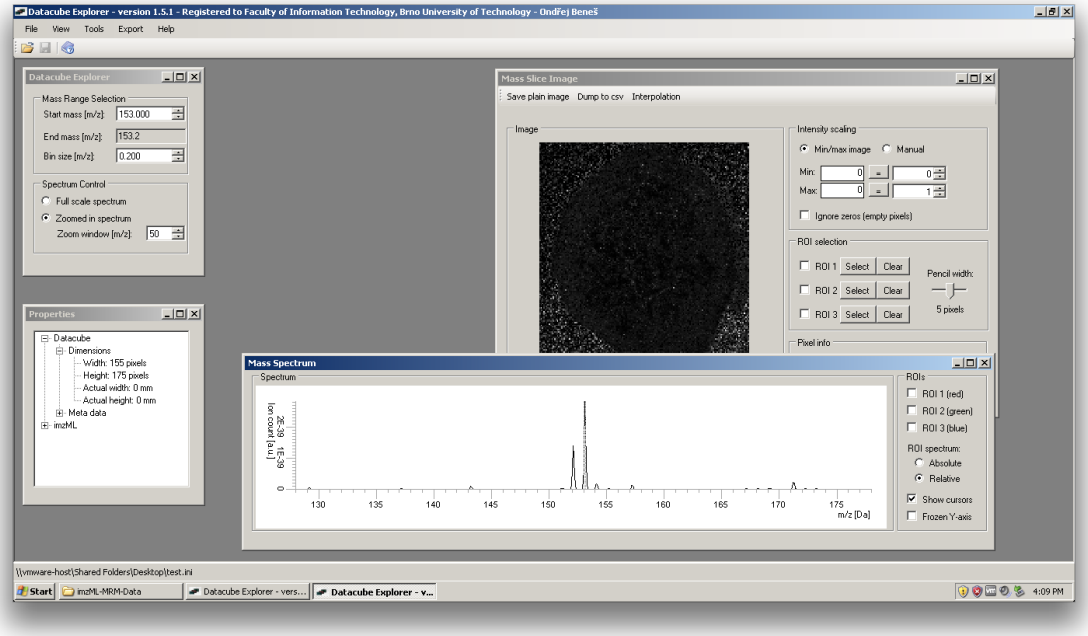
Aplikace Datacube Explorer (obrázek 4.2) je jedna z řady aplikací, které jsou schopné vizualizovat data zobrazovací hmotnostní spektrometrie. Je však jednou z mála, které umí vizualizovat data z imzML souborů, BioMap souborů a navíc ještě z formátu datacube, který byl vytvořen za účelem pokročilejší vizualizace ve 3D pomocí dodatečného software. Tento formát byl vytvořen právě při tvorbě této aplikace. Aplikace byla vytvořena pro jednoduché a rychlé zobrazení velkých souborů zobrazovací hmotnostní spektrometrie. Datacube Explorer nabízí základní jednotný způsob vizualizační metody pro data zobrazovací hmotnostní spektrometrie, který je výstupem různých přístrojů a přístupů, což z něj činí nezávislým na poskytovateli těchto dat (obrázek 4.1).

Datacube Explorer je nejpoužívanější software v oblasti zobrazovací hmotnostní spektrometrie pokud se pracuje se soubory ve standardu imzML [12]. Pro následnou vizualizaci ve 3D se používá další software s názvem Volume Explorer. Mimo samotné vizualizaci software s daty hmotnostních spekter nic nedělá, zaměřuje se čistě na vizualizaci, neprobíhá zde žádné předzpracování dat. Pokud je tedy u dat například vysoký šum, je nutné tyto vstupní data předzpracovat před samotným zobrazením v jiném software. Aplikace umí jednoduše barevně zvýraznit takzvané oblasti ROI (z anglického region of interest).

Datacube Explorer dostupný ke stažení zdarma, avšak jen v binární podobě. Pro správnou funkci programu je ještě nutné autory požádat o licenci, bez které program není schopen načítat jakékoliv soubory. Program byl vytvořen v prostředí Visual Studio v programovacím



Obrázek 4.1: Schéma zpracování dat v programu Datacube Explorer pro vizualizaci [12]



Obrázek 4.2: Rozhraní aplikace Datacube Explorer

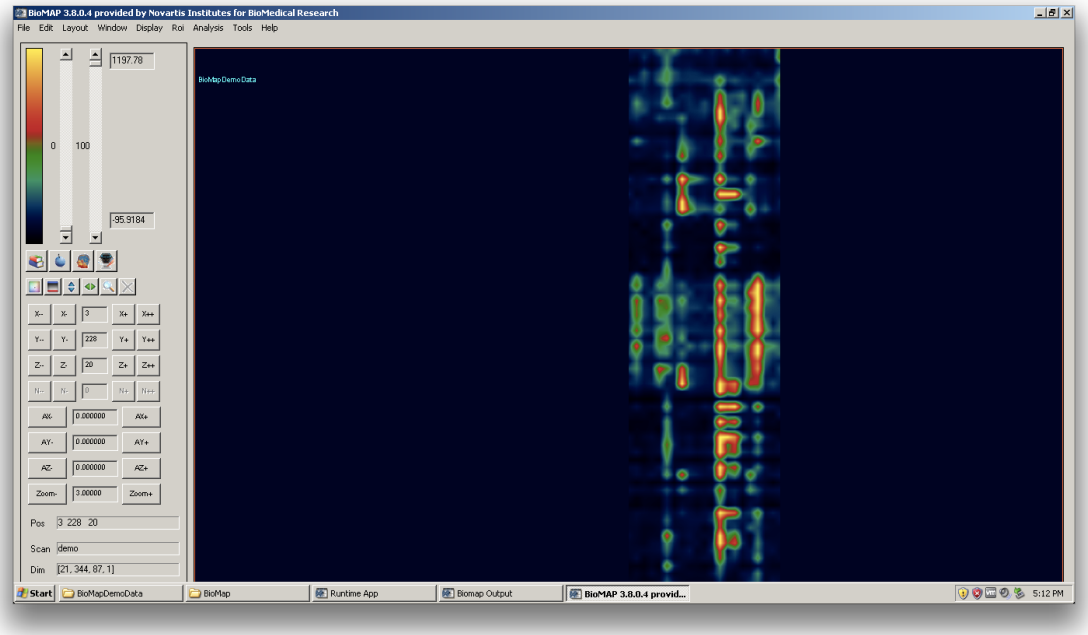
jazyce `C#`, díky čemuž je spustitelný pouze na systémech Microsoft Windows s nainstalovaným .NET framework ve verzi 4 a vyšší. Jeho zdrojový kód však není veřejně k dispozici a aplikace se dá považovat za uzavřenou. Zdrojový kód je dostupný jen pro software pro následnou vizualizaci ve 3D, která probíhá pomocí již zmíněného nástroje Volume Explorer. Vývoj na této aplikaci je však aktivní a během psaní této práce vyšlo hned několik jeho vedlejších verzí a také jedna nová hlavní verze.

BioMap

Aplikace BioMap (obrázek 4.3) byla původně v roce 1996 vyvinuta pro vyhodnocování MRI[1] dat z biomedicínských výzkumů. V současné době podporuje i jiné zobrazovací techniky

[1] Magnetic resonance imaging - medicínská zobrazovací technika používaná v radiologii ke zjištění anatomie a funkce lidského těla

včetně zobrazovací hmotnostní spektrometrie. Samotná aplikace byla napsána v IDL™[2]. Díky tomu lze aplikaci spouštět na různých systémech a také ji různě rozšiřovat pomocí softwarových balíčku individuálně zaměřených na jednu konkrétní problematiku. Vizualizace je založena na rekonstrukci více rovin umožňující extrakci libovolných řezů z volumetrických dat. BioMap používá čtyřrozměrnou reprezentaci dat pro reprezentaci obrázku. První tři rozměry popisují pozici samotného voxelu[3] v prostoru a čtvrtá dimenze je vyhrazena pro nezávislý parametr jako například čas, hmota či vlnová délka. Samotná data lze do BioMap načíst v několika formátech, standard imzML však aplikace v současné době nepodporuje. Podpora pro tento formát je v současné době ve vývoji. Aplikace je schopna výsledný obrázek různě zpracovávat, ale na úrovní obrazových dat určených spíše pro MRI. Předzpracování dat z hmotnostních spektrometrů jak bylo popsáno v kapitole 5.5 zde není dostupné. Visuálně analyzovaná data lze uložit do formátu dat, který lze dál zpracovávat v dalším software. BioMap zahrnuje i další nástroje, které jsou vyžadovány pro zpracování konkrétních dat, umožňuje zkombinovat výsledky z několika různých dat či experimentů a zdokumentovat výsledek studie [22]. Program je volně ke stažení v binární podobě, pro běh na konkrétní platformě je nutné nejprve stáhnout interpret IDL. Pro plnou funkčnost je nutné aplikaci online licencovat. Ani autoři této aplikace zdrojový kód nezveřejňují.



Obrázek 4.3: Ukázka hlavního okna aplikace BioMap se vzorovými daty

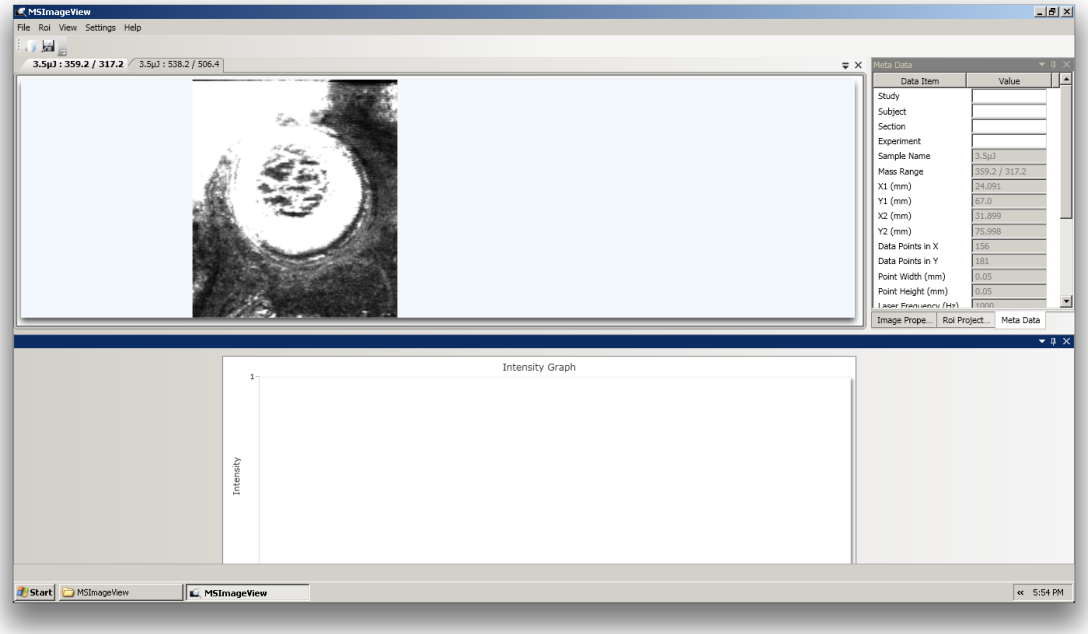
MSImageView

Program MSImageView (obrázek 4.4) je dílem autorů podílejících se na aplikaci BioMap 4.1. Původním cílem této aplikace byla konverze dat získaných pomocí přístroje FlashQuant do

[2]Interface Description Lagungage - jazyk popisující výsledné rozhraní dle komponent nezávisle na programovacím jazyce

[3]volumetrický pixel

obrázku zobrazovací hmotnostní spektrometrie [23]. Aplikace umí tedy pouze číst a zobrazit soubory `wiff` vytvořené na tomto přístroji. Umí zobrazit graf intenzity a základní informace o datech. Zobrazená data umí uložit ve formě klasických obrázku PNG, JPEG či TIFF nebo do formátu imzML. Nejvíce se tato aplikace podobá aplikaci Datacube Explorer s tím, že umí pouze číst data `wiff`. Aplikace je volně ke stažení a od začátku roku 2014 je k ní i volně dostupný zdrojový kód v jazyce `C#`. Z toho vyplývá, že aplikaci je možné spustit jen na systémech Microsoft Windows.



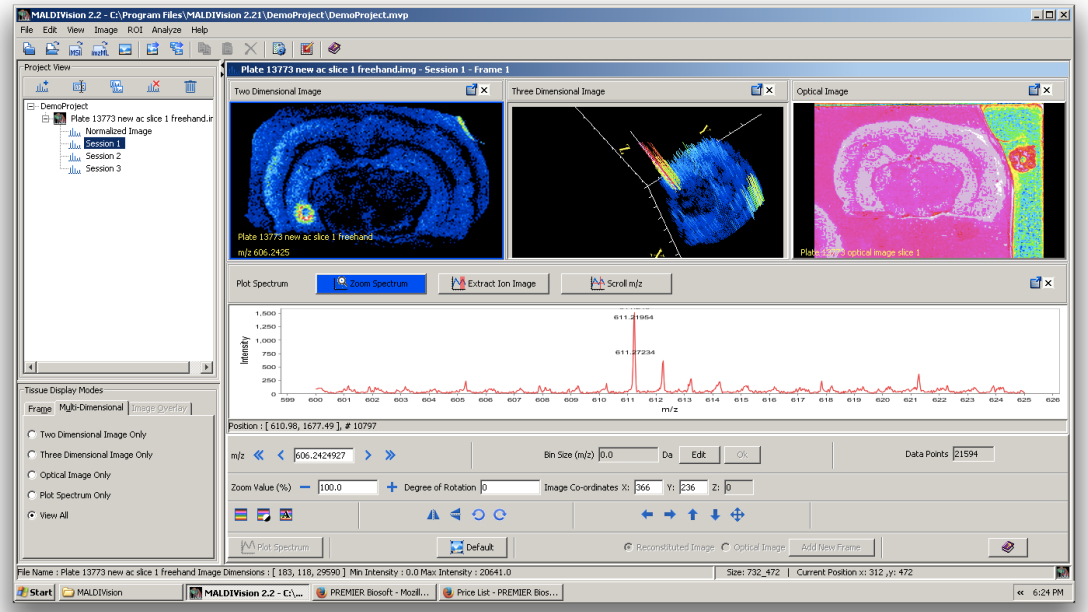
Obrázek 4.4: Uživatelské rozhraní aplikace MSImageView

MALDIVision

Komerční software MALDIVision (obrázek 4.5) pochází od firmy PREMIER Biosoft. Jedná se o plnohodnotný vizualizační nástroj pro zobrazovací hmotnostní spektrometrii. Program umí načítat a zobrazovat data ze standardu imzML. Umí zobrazit až 10 obrázků současně, kde každý obrázek zobrazuje různý kontext či dokonce různé měření. Má široké možnosti nastavení výstupního obrázku ať už se jedná o barevné schéma či o zobrazení pouze podmnožin měření. Lze v něm dokonce i načíst fotografii zkoumaného vzorku pro porovnání s hyperspektrálním obrázkem. Má také jednoduché zobrazení grafu spektra. Jelikož je software komerční a jeho licence vyjde na necelých $10000 byla v průzkumu ozkoušena pouze demoverze, která neumožňuje načítat jiná než ukázková data.

MSiReader

MSiReader (obrázek 4.6) je open source vizualizační nástroj postavený na platformě MATLAB. MSiReader chce být alternativou pro vědce, kteří chtějí prozkoumávat vlastní zdroje obrázků a mít pod kontrolou zpracování jejich dat. Jako vstup aplikace podporuje mzXML,



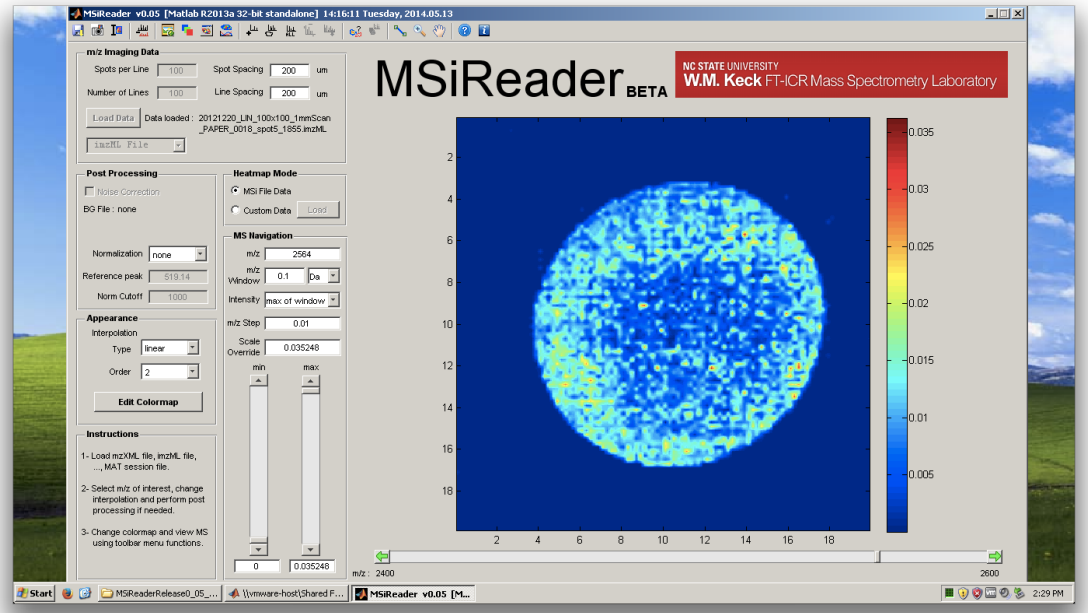
Obrázek 4.5: Zobrazení tří různých pohledů na data z ukázkového experimentu společně s náhledem konkrétního spektra

imzML, img či ASCII. Klade si za cíl umožnit otevírání obrázků, bez jakýchkoliv limitů velikostí či hojností naměřených dat. Aplikace po otevření ve spektru pomocí automatického vyhledávání nalezne píky, na jejichž základě zobrazí úvodní hyperspektrální obrázek. V aplikaci lze provádět základní předzpracování jako jsou korekce základny či normalizace. Bohužel aplikace neobsahuje možnost procházení a ani zobrazení jednotlivých spekter a musí se zde spoléhat na automatické vyhledávání na začátku. Je obtížné v aplikaci najít intenzity signálů v neznámem vzorku pro zobrazení správného obrazce. [18]

Aplikace je k dispozici jako samostatná aplikace, kterou lze spustit v MATLAB runtime. V současné době je aplikace ve verzi 0.05 a je ve fázi vývoje. Autoři s aplikací rovněž distribuují zdrojové kódy pro prostředí MATLAB a je tedy možné aplikaci spustit přímo v tomto prostředí.

flexImaging

Německá společnost Bruker zabývající se výrobou hmotnostních spektrometrů ke svým přístrojům dodává celou řadu software. Jedním z nich je i software pro vizualizaci zobrazovací hmotnostní spektrometrie s názvem flexImaging (obrázek 4.7). Tento software umí pracovat s daty zobrazovací hmotnostní spektrometrie ve speciálním formátu, který aplikace získá přímo z hmotnostního spektrometru skrze další aplikaci s názvem flexControl. Přímo z této aplikace lze tedy rovnou spouštět experimenty. Data z výstupu experimentu jsou pak rozdělena po jednotlivých spektrech, tedy každý změřený bod má vlastní soubor, který je umístěn ve vlastním adresáři. Jedno takové měření může pak obsahovat například 70 tisíc samostatných souborů. Tyto soubory aplikace při otevření přednačítá pravděpodobně do nějaké interní databáze neboť po otevření nedochází k dodatečnému čtení a software je



Obrázek 4.6: Hlavní okno aplikace MSiReader

schopen otevřít i velké soubory s velikosti kolem 4GB. Načítání však trvá několik desítek minut, ale práce s těmito daty je pak plynulá.

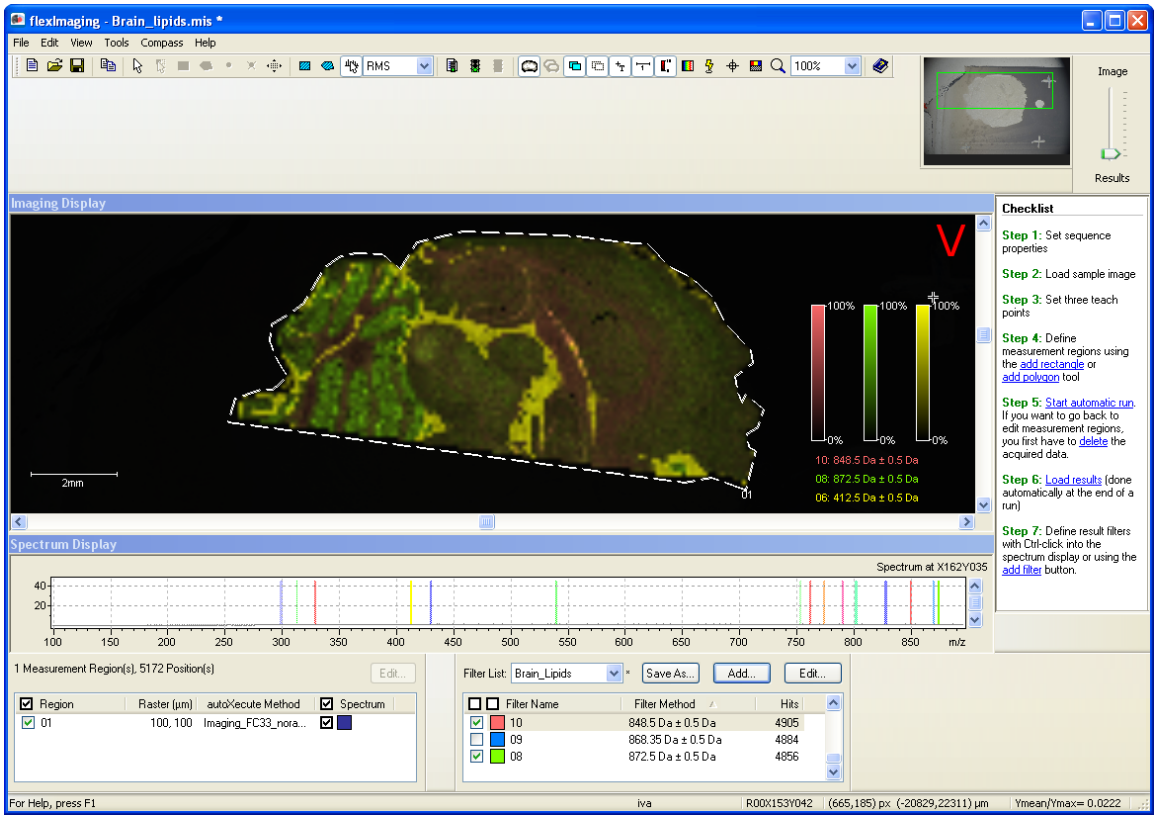
Aplikace umí zobrazit jeden hyperspektrální obrázek. Tento obrázek je možné překrýt s reálnou fotkou a tyto obrázky prolínat mezi sebou. Ve spektru lze v aplikaci zvolit několik různých oblastí, které se následně barevně odliší a ve výsledném obrázku se to projeví vykreslením různých oblastí různě barevně. V obrázku lze také označit oblasti ROI a aplikace pak pracuje pouze nad těmito oblastmi.

FlexImaging samotný umí jediný krok předzpracování a to normalizaci. O ostatní kroky se stará ostatní software z jejich nabídky. Například vyrovnání základny či redukci dat řeší software flexControl již při měření. Pokročilejší analýzy nad daty pak provádí software Data Analyzis.

Software bohužel neumí pracovat s daty ve formátu imzML. Není také dostupný volně ke stažení neboť je dodáván přímo s přístrojem. Ani jeho zdrojový kód není dostupný.

mMass

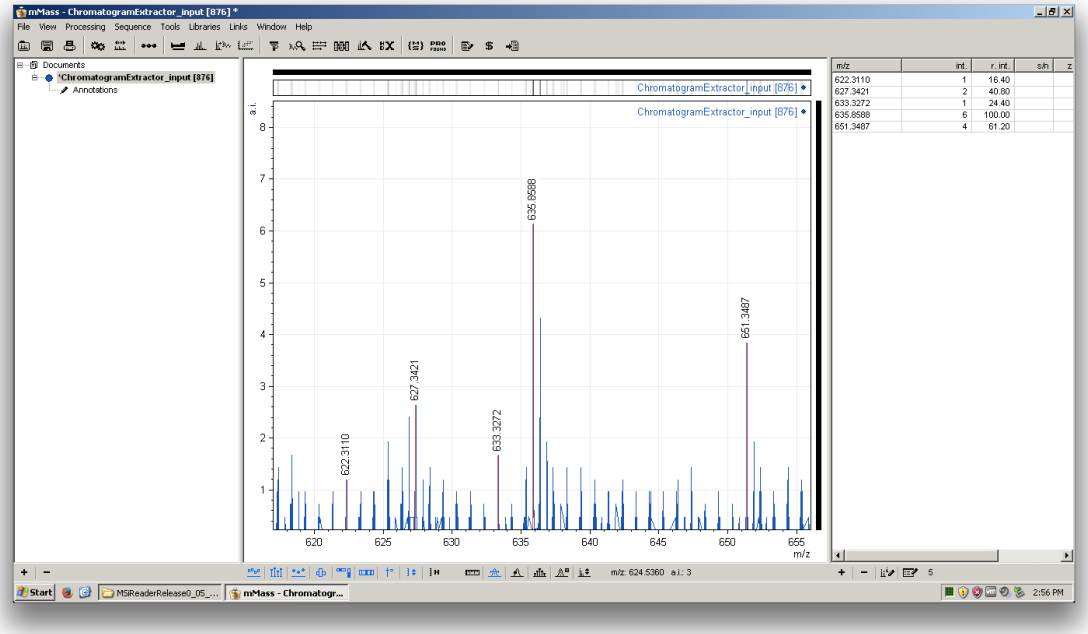
Open source aplikace mMass (obrázek 4.8) je aplikace napsaná v jazyce Python vývojářem Martinem Strohalmem. Aplikace je kompletní nástroj pro práci s hmotnostními spektry. Obsahuje mnoho nástrojů od předzpracování dat jako je kalibrace, korekce základny, vyhlazování či dekonvoluce, přes nástroje umožňující porovnání samotných spekter, jejich hromadné zpracování až po různé nástroje pro práci v rámci proteomiky jako je identifikace a evidence proteinů. [24] Bohužel aplikace v současné době neumí pracovat se soubory imzML a ani s jinými soubory zobrazovací hmotnostní spektrometrie. Vývoj za účelem přidání podpory pro tento typ dat je nyní ve vývoji.



Obrázek 4.7: Hyperspektrální obrázek krysího mozečku zobrazený v aplikaci flexImaging

4.2 Shrnutí

Aplikace, které jsou v současné době k dispozici pro práci s daty v rámci zobrazovací hmotnostní spektrometrie sice existují, ale jejich možnosti nejsou velké. Většina aplikací umí zobrazit hypespektrální obrázek z určitého konkrétního formátu dat, který je často různý, není standardem v odvětví a nelze jej jednoduše sdílet. Standard imzML 3.4, který se snaží sjednotit tento formát podporuje jen malé množství aplikací. Aplikace s otevřeným zdrojovým kódem, která by uměla tento nový formát dat přečíst a umělá zároveň data ze zobrazovací hmotnostní spektrometrie předzpracovat, prakticky na trhu zatím neexistuje a je tedy nutné se spoléhat na komerční software, který je buď prodáván k samotným hmotnostním spektrometrům či samostatně. Je ale vidět, že vývojáři jednotlivých programů se snaží do většiny aplikací nový standard doplnit.



Obrázek 4.8: Aplikace mMass umí pracovat pouze s jednotlivými spektry. Ukázka hlavního okna s jedním takovým spektrem ze souboru mzML

Kapitola 5

Návrh a implementace

V následující kapitole je nejprve shrnuto, co by výsledná aplikace měla umět. V potaz se berou jak výstupy z předchozí kapitoly o současném stavu, tak požadavky, které vzešly z diskuzí přímo v laboratoři Masarykovy univerzity s vědci, kteří pracují s hmotnostním spektrometrem z kapitoly 2.6. Na základě těchto požadavků byla navržena struktura aplikace blíže popsána v sekci 5.2. Dále je pak v sekci 5.3 popsán způsob a implementace načítání dat z imzML. V sekci 5.4 je popsáno grafické uživatelské rozhraní vzhled aplikace a stručně je popsán hlavní návrhový vzor 5.4. A v neposlední řadě je přiblížena implementace jednotlivých procesů předzpracování v sekci 5.5.

5.1 Požadovaná funkčnost

Tato práce probíhala ve spolupráci s Laboratoří bioanalytické instrumentace prof. Jana Preislera. V rámci této laboratoře byl vyroben a sestaven vlastní hmotnostní spektrometr. Kvůli tomu byl k tomuto přístroji nejprve vyvinut ovládací program, který umožňuje ovládat příslušné komponenty přes rozhraní počítače [13]. Tento software data z experimentů ukládá přímo do formátu imzML (3.4).

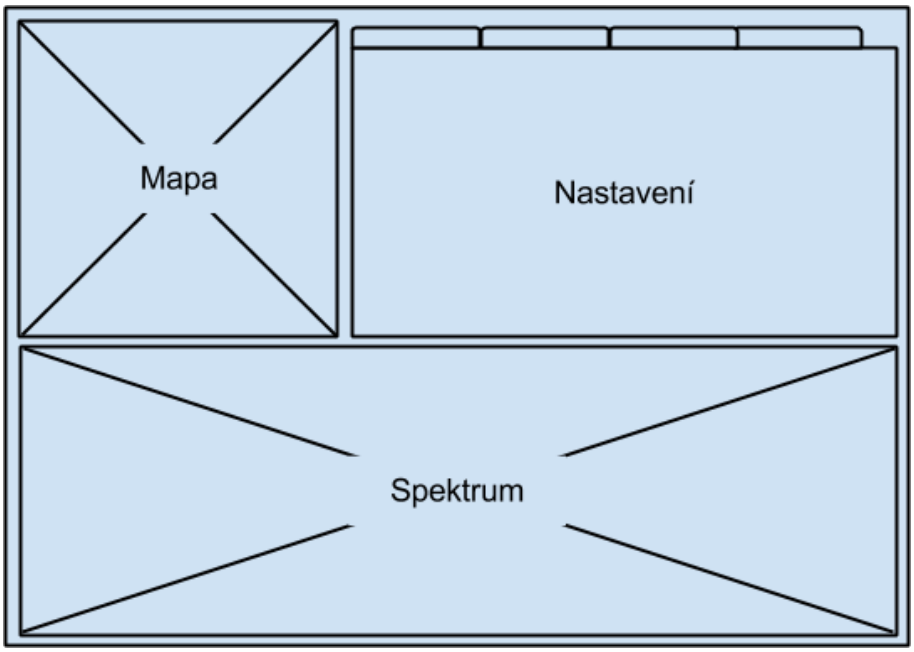
Tyto data je však nutné nadále otevřít a zpracovat, což již zmiňovaný ovládací program neumí. První z požadavků je tedy aby aplikace uměla pracovat se standardem imzML. Z toho vyplývá, že je nutné, aby aplikace uměla pracovat s daty zobrazovací hmotnostní spektrometrie. Je požadováno, aby aplikace byla schopna zobrazit spektrum pro každý bod měření a to buď výběrem bodu v mapě intenzit nebo výběrem konkrétního spektra ze seznamu všech spekter. Toto spektrum bude obsahovat spousty bodů a některé informace nemusí být v takovém přehledu zřejmé či se ztratí. Je tedy nezbytné, aby bylo možné zobrazovat části takového spektra zvlášť neboli mít možnost přiblížení či oddálení. Aplikace by měla také umět zobrazit průměrné spektrum. Při zobrazení spektra by mělo být možné vybrat jeden konkrétní bod a případně jeho okolí neboli interval okolo tohoto bodu. Na základě tohoto bodu by měla aplikace umět vykreslit 2D mapu intenzit v celém měření. Tento bod by měl jít vybrat i v průměrném spektru.

Ani měření na vlastním hmotnostním spektrometru se nevyhne rušivým vlivům, a proto je požadováno, aby aplikace uměla základní kroky předzpracování dat spektra. Jedním z důležitých kroků je kalibrace, jenž je popsána v kapitole 5.5. Pro účely laboratoře je požadováno, aby aplikace měla možnost interní kalibrace, tedy měla možnost výběru několika bodů, ke kterým se doplní jejich očekávané hodnoty a aplikace na základě těchto hodnot provede samotnou kalibraci. Měly by být k dispozici minimálně kalibrace lineární či kvad-

ratická. K jednodušší kalibraci by aplikaci mohl sloužit nějaký algoritmus, který by dokázal body ke kalibraci nalézt sám. Dále by aplikace měla umět vyhladit měřená spektra, jak je popsáno v kapitole 5.5. Důležitá je implementace vyhlazovacího filtru Savitzky-Golay, která je v tomto odvětí nejvíce používána. Další techniky předzpracování jako je korekce základny pro potřeby laboratoře nejsou nutně vyžadovány. Výstupy z přístroje totiž nevykazují deformaci základny. Aplikace by však měla být v budoucnosti rozšířitelná o tento či další kroky předzpracování.

5.2 Návrh struktury aplikace

Samotná aplikace bude vyžadovat grafické uživatelské rozhraní a to z důvodů vizualizace spekter či 2D mapy intenzit. Nejprve je tedy nutné stanovit, jaké budou hlavní komponenty uživatelského rozhraní. Návrhy a nápady vzešly z diskuzí s vědeckými pracovníky a z průzkumu současného stavu, který je přiblížen v kapitole 4. Výsledkem byl návrh uživatelského rozhraní, jak je uveden na obrázku 5.1.



Obrázek 5.1: Návrh rozložení komponent uživatelského rozhraní výsledné aplikace

Mapa intenzit

U většiny aplikací pro zobrazovací hmotnostní spektrometrii je hlavní komponenta samotná 2D mapa intenzit. Některé aplikace mají těchto map možnost zobrazit více, jiné zobrazují pouze jednu. Uživatelské rozhraní by tedy mělo minimálně jednu takovou mapu zobrazovat neboť je to základní kámen pro zobrazovací hmotnostní spektrometrii. Tato mapa by měla zobrazovat různé intenzity různou barvou. Většina aplikací má také možnost zvolit různou barevnou mapu pro tyto intenzity. Některé informace se mohou pro lidské oko v jedné barevné škále ztratit a v jiné zase ukázat, a proto by bylo vhodné kdyby aplikace uměla tyto barevná schémata také měnit. Mapa by měla mít možnost výběru konkrétního bodu

a samozřejmě indikace takového výběru. Okno grafického uživatelského rozhraní by tedy mělo zobrazovat takovou mapu jako jednu z hlavních komponent.

Spektrum

Většina z testovaných aplikací neměla hmotnostní spektrum jako jednu z hlavních komponent, případně nebyl vůbec přítomná. Kvůli potřebám operátorů přístroje je vhodné mít komponentu se zobrazeným spektrem na stejné úrovni jako zobrazenou mapu intenzit. Zobrazit spektrum lze nejlépe pomocí grafu, kde na ose X budou umístěny hodnoty m/z neboli jednotky Dalton a na ose Y hodnoty intenzity. Ač jsou hodnoty měření diskrétní je vhodné spojit jednotlivé hodnoty po sobě jdou hodnoty intenzity čárou neboť pak je jednoduší visuálně analyzovat průběh funkce spektra.

Nastavení předzpracování

Každý krok předzpracování má specifické nastavení či vyžaduje zobrazit specifická data. Je tedy nutné mít rozhraní pro jednotlivé kroky. Jako vhodné se nabízí seskupit tyto části do jednotlivých sekcí do jedné komponenty, která umožní přepínaní mezi těmito komponentami. Přepínání mezi kontextem by pak mělo probíhat pomocí přepínačů či záložek.

5.3 Zpracování imzML

Jak již bylo uvedeno v kapitole 4 formát imzML je standard vycházející ze standardu mzML. U mzML bylo celé spektrum uloženo přímo v samotném souboru. Oproti tomu formát imzML samotná naměřená data neobsahuje a data dělí do dvou souborů, jeden který je zpravidla menší a obsahuje pouze metadata a druhý, který obsahuje data samotná.

Čtečka hyperspektrálních obrázků je implementována v jazyce Ruby a balíčkovací systém pro tento jazyk RubyGems žádný balíček pro práci s tímto formátem neobsahuje. Bylo nutné takový balíček nejprve vytvořit. Balíček byl vytvořen jako součást této práce a má za cíl implementovat čtení a popřípadě i zápis souborů imzML. Soubor s metadaty je v jádru XML a jelikož cílem této práce není sestrojení parseru, byla použita externí knihovna pro práci s s XML/HTML s názvem Nokogiri. XML se obecně zpracovává dvěma způsoby a to pomocí DOM či SAX parserů. První z nich načte celé XML do paměti a vytvoří takzvaný DOM strom, který obsahuje všechny data ze souboru XML strukturovaně. Druhý zmiňovaný přístup SAX zpracovává XML postupně po každém uzlu a nenačítá celé XML do paměti. Data se v tomto přístupu získávají skrze události, které jsou při každém čtení jednotlivých uzlů vyvolány.

Nejdříve byly data zpracovány pomocí DOM neboť tento přístup nevyžaduje řešit vnořování jednotlivých elementů XML souboru. Na testovacích souborech tato metoda fungovala spolehlivě, ale ve chvíli přišla na řadu reálná data z běžně používaného přístroje se tento přístup ukázal být nevhodný. I přestože jsou samotná data od měření v samostatném souboru, měření v zobrazovací hmotnostní spektrometrii může vykazovat velké množství metadat. U reálných dat tak výsledný soubor s metadaty může dosahovat i velikosti desítek MB. Zde se tedy ukázalo použití metody SAX jako vhodnější neb je tento přístup o něco rychlejší. Bohužel knihovna Nokogiri ani s metodou SAX nedosahovala optimálních výsledků, byla tedy zvolena optimalizovaná, druhá avšak méně známá knihovna ox, která dosahuje mnohem lepších výsledků.

Celým jádrem balíčku pro zpracování imzML souboru jsou dvě hlavní třídy `ImzML::Parser` a `ImzML::Metadata`. První z nich zajišťuje správné zpracování souboru dle standardu a výsledného použití. Konstruktor přijímá pouze dva parametry a to cestu k souborům imzML a ibd, přičemž druhý zmiňovaný je nepovinný a v případě jeho nevyplnění se automaticky předpokládá, že binární soubor je uložen a pojmenován stejně jako imzML soubor vyjma koncovky. Parser není pouze náhradou za DOM přístup, ale rovnou bere v potaz výslednou sémantiku dat pro jednoduší práci s daty. V základu parseru figuruje automat, který za pomocí dvou zásobníků, jeden pro zanoření, druhý pro samotné elementy, prochází celý soubor postupně a spojuje jednotlivé elementy metadat, které spolu souvisí. Výsledná data po parsování jsou uloženy v atributu s názvem metadata, který je instancí již zmiňované třídy `ImzML::Metadata`. Tato třída obsahuje několik atributů, které podobně jako samotná metadata obsahují informace o samotném experimentu, použitých přístrojích či metod pro získání dat. V neposlední řadě také obsahuje atribut spectrums, který obsahuje instanci objektu Hash, která obsahuje metadata k jednotlivým skenům. Každý takový sken je pak implementace třídy `ImzML::Spectrum`. Tato třída obsahuje metody a atributy pro získání samotných dat z měření bez ohledu na metodu původního měření (nepřetržitý nebo zpracovaný). Atributy `mz_binary` a `intensity_binary` implementují třídu `ImzML::Spectrum::BinaryData` a zajištují přístup k samotným datům. Třída `ImzML::Spectrum::BinaryData` umí tyto načítaná data ukládat do paměti, aby se nemusely pokaždé načítat znovu.

V imzML souboru se hojně používají hodnoty z kontrolovaného slovníku, které ukazují do různých OBO souborů. Určité identifikátory jsou přímo definovány ve standardu, proto jsou v příslušných třídách uloženy jako konstanty. Díky rozšiřitelnosti OBO formátu mohou být některé parametry v budoucnu přidány, a proto se kontrolují dynamicky na základě jejich pozice ve stromové struktuře uvedené v OBO souborech. Za tímto účelem jsou v imzML souboru elementy *cv*, které popisují, jaké OBO souboru jsou použity a to včetně odkazů na webové stránky, kde jdou stáhnout. Protože tyto odkazy v některých případech nebyly funkční, byly tyto soubory přibaleny k balíčku. Ke změnám v těchto souborech dochází zřídka jen na základě rozhodnutí příslušných konsorcií.

Balíček imzml obsahuje automatické testy pokrývající důležité části zpracování imzML souboru.

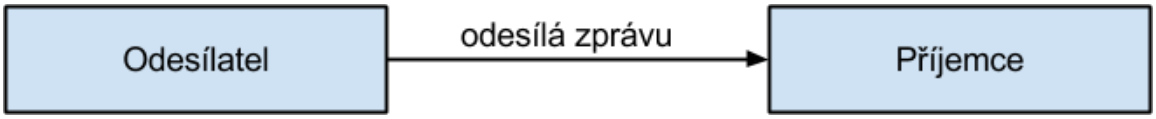
5.4 Grafické uživatelské rozhraní

Ruby je vysoce úrovňový objektově orientovaný skriptovací jazyk, kde je všechno s čím se pracuje objekt, dokonce i výsledky jakýchkoliv operací jsou objektem. Základní vlastnosti jazyka lze rozšířit balíčky nazývanými RubyGems. Díky tomu, že je jazyk interpretovaný, lze jej spustit na různých virtuálních strojích. Jazyk je také multiplatformní. Hlavní výhodou tohoto jazyka je možnost časté iterace výsledné aplikaci díky vysoké abstrakci jednotlivých operací [26]. Tento jazyk byl zvolen jako implementační jazyk pro tvorbu aplikace v této práci. Pro tvorbu uživatelského rozhraní byl zvolen balíček FXRuby.

FXRuby

Knihovna FXRuby je samotná jen obal neboli wrapper pro C++ rozhraní knihovny FOX toolkit, která zajišťuje samotné vykreslování jak samotných oken, tak jednotlivých prvků uživatelského rozhraní. Knihovna FXRuby využívá event-driven styl komunikace. Po inicializaci program v FXRuby vstoupí do takzvaného cyklu událostí (event loop), kde program

očekává až nastanou nějaké události. Každá taková událost je zde reprezentována odesílatelem, který odesílá zprávu příjemci (obrázek 5.2). [9]



Obrázek 5.2: Ukázka základní komunikace v knihovně FXRuby

Každá zpráva v FXRuby se skládá z typu zprávy, identifikátoru zprávy a daty vztahující se ke zprávě. Typ zprávy je daný konstantou začínající řetězcem `SEL_` například `SEL_COMMAND`. Identifikátor zprávy je také konstanta, kterou příjemce používá k rozlišení dvou příchozích zpráv stejného typu. Událost může volitelně posílat také různé objekty jako data. Kód, který se má vykonat po přijetí zprávy se pak standardním způsobem v Ruby přiřadí pomocí bloku kódu neboli instancí třídy `Proc`. Ukázka použití:

```
button.connect(SEL_COMMAND) do |sender, selector, data|
 # kód vykonaný po zmáčknutí tlačítka
end
```

Rozložení prvků v okně aplikace v FXRuby mají na starosti objekty `FXPacker` neboli takzvané layout manažery. Každému takovému objektu se v konstruktoru nastaví vlastnosti jako jsou zarovnání, roztahování či odsazení. Knihovna FXRuby obsahuje několik tříd, které dědí z objektu `FXPacker` a dál jeho chování generalizují. Tyto objekty lze do sebe vnořovat.

Model-view-controller

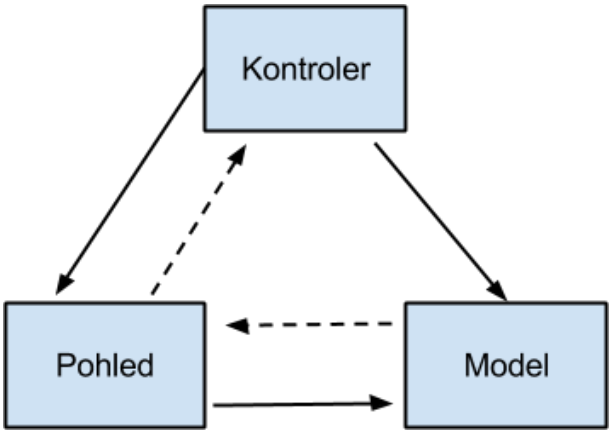
Nad základnímy principy FXRuby byl v rámci projektu použit návrhový vzor Model-view-controller (dále jen MVC). Tento návrhový vzor je používaný od dob vzniku jazyka Smalltalk. Je to vysoce úrovňový návrhový vzor, který zajímá globální architektura aplikace a klasifikuje objekty do rolí, které v aplikaci hrají. Aplikace využívající tento návrhový vzor jsou jednoduší na úpravy a změny požadavků tedy lépe rozšiřitelné. MVC uvažuje tři základní typy objektu (obrázek 5.3). Objekty modelu, objekty pohledu (z anglického *view*) a objekty kontrolérů (z anglického *controller*). Návrhový vzor těmto objektům definuje roli v celkové komunikaci aplikace. [2]

Model

Objekty ze skupiny model reprezentují určitou znalost či odbornost. Drží a definují logiku, která manipuluje s daty. Díky tomu, že jsou znalosti či odbornosti vztažené ke konkrétní problematice, mají tendenci být opakovaně použitelné. V ideálním případě nemá objekt modelu žádné spojení s uživatelským rozhraním.

Pohled

Objekt ze skupiny pohled ví, jak má zobrazit uživateli data z objektu modelu. Tyto objekty by však neměli být zodpovědné za ukládání zobrazených dat. Mohou zobrazovat takových objektů více nebo jen jejich částí.



Obrázek 5.3: Schéma komunikace návrhového vzoru Model-view-controller

Kontrolér

Kontroléry, objekty z poslední skupiny, jsou pak v aplikaci zprostředkovateli mezi objekty ze skupiny pohled a objekty ze skupiny model. Často tyto objekty zajišťují, aby pohled zobrazoval správný model, případně aby se do modelu ukládala správná data.

Objektový návrh UI

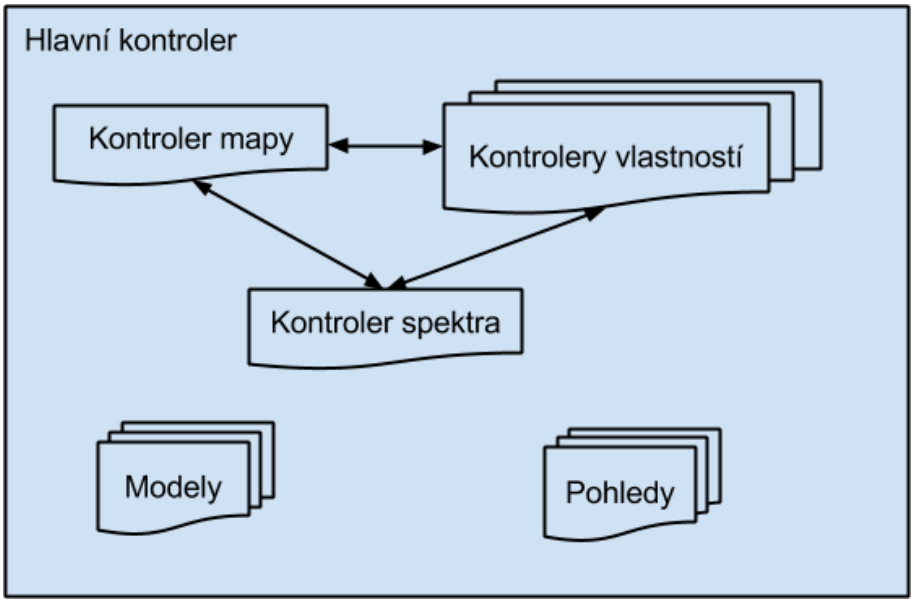
Na základě návrhu komponent uživatelského rozhraní v sekci 5.2 byl jako základ zvolen objektový návrh MVC. Celé uživatelské rozhraní představuje jeden modul s názvem `Hyperspectral`. Hlavní okno je reprezentováno třídou hlavního kontroléru `MainController`. Tato třída zajišťuje základní funkce hlavního a jediného okna aplikace. Zároveň zajišťuje načítání ostatních kontrolérů. Tyto kontroléry lze rozdělit do tří částí. První z nich je kontrolér starající se o mapu intenzit `ImageController`, další je kontrolér pro správu zobrazení spektra `SpectrumController` a dále jsou ostatní takzvané kontroléry vlastností, kde každý kontrolér má na starosti jednu konkrétní část předzpracování. Tyto tři skupiny kontrolérů mezi sebou vzájemně komunikují a předávají si data. Dále jsou přítomny objekty modelů a pohledu, které komunikují se samotnými kontroléry. Vše je schematicky zobrazeno na obrázku 5.4 Každý kontrolér má na starosti nějakou část uživatelského rozhraní, a proto obsahuje metodu nazvanou `load_view`, která obsahuje kód pro vytvoření rozhraní pro daný kontrolér. Metoda má jediný parametr nazvaný `superview`, do kterého se předává objekt uživatelského rozhraní, který bude jeho rodičem a všechno uživatelské rozhraní se bude vytvářet uvnitř.

Zpětné volání

Vzájemná komunikace mezi jednotlivými kontroléry je provedena pomocí vlastního systému zpětného volání (anglicky callbacks). Ruby jako jazyk disponuje možnostmi metaprogramování. To znamená, že je za běhu možné měnit samotný kód aplikace. Další intenzivně používanou vlastností jazyka Ruby je schopnost pracovat s bloky kódu jako objekty [16]. Tyto dvě vlastnosti byly využity k vytvoření systému zpětného volání.

Samotná logika zpětného volání je implementována v modulu `Callbacks`. Tento modul obsahuje jeden privátní atribut `callbacks` a tři metody:

- `method_missing(m, *args, &block)`



Obrázek 5.4: Upravený objektový návrh MVC aplikace

- `callback(name, *args)`
- `callback_exists?(name)`

První a hned nejdůležitější metoda je metoda, kterou Ruby zavolá v případě, že daný objekt volanou metodu neimplementuje. Jako první argument je předán objekt `Symbol` obsahující název volané metody. Zbylé dva parametry obsahují argumenty předané volané metodě a popřípadě i blok kódu. Metoda při prvním přístupu inicializuje atribut `callbacks` jako instanci objektu `Hash` neboli asociativního pole. Následně a i při každém dalším přístupu do tohoto pole na pozici rovnající se názvu metody uloží daný blok kódu.

Druhá důležitá metoda má název `callback`. Jako první parametr se metodě předává název, který může být následován argumenty. Tato metoda se nejprve podívá, zda v atributu `callbacks` existuje metoda s názvem, který byl předán jako první parametr a pokud ano, tak vykoná blok kódu s uloženy na pozici pro tuto metodu s argumenty z parametrů metody. Poslední metoda `callback_exist?` je jen pomocná metoda pro zjištění existence zpětného volání.

Použití tohoto způsobu komunikace reflektuje zvyklosti v jazyce Ruby a je podobný způsobu komunikace event-driven v balíčku FXRuby. Nejprve požadovanou třídu rozšíříme pomocí `include` o metody modulu `Callbacks`. Jako příklad použití si uvedeme situaci, kdy v hlavním okně přes menu uživatel vyvolá otevření souboru a vybere nějaký soubor. Po provedení této události by se měl v aplikaci otevřít a načíst zvolený soubor. Mějme tedy instanci objektu `menu`, u kterého zavoláme metodu `when_file_opens` následovně

```
menu.when_file_opens do |filepath|
  open_file(filepath)
end
```

Kód pro otevření souboru se uloží do atributu `callbacks` a je vyvolán ve chvíli kdy menu dokončí výběr souboru. Menu vyvolá tento kód pouhým zavoláním metody `callbacks`, kterou díky rozšíření o modul `Callbacks` implementuje. Příklad použití:

```
Fox::FXMenuCommand.new(file_menu, 'Open...').connect(Fox::SEL_COMMAND) do
  dialog = Fox::FXFileDialog.new(self, 'Open imzML file')
  dialog.directory = DEFAULT_DIR
  dialog.patternList = ['imzML files (*.imzML)']

  if (dialog.execute != 0)
    callback(:when_file_opens, dialog.filename)
  end
end
```

Dlouho trvající operace

V aplikaci často dochází k časově náročným operacím a je třeba uživateli nějakým způsobem dát vědět o stavu operace. Takové operace je vhodné pouštět na pozadí, aby nedocházelo k úplnému zablokování uživatelského rozhraní. K tomu je v hlavním okně aplikace objekt třídy `ProgressDialog`, který dědí z objektu `Fox::FXProgressDialog`. Objekt je inicializován v hlavním okně, kde se na něm volá metoda `run` s blokem kódu. Tento blok je v tomto objektu spuštěn na novém vlákně a sám objekt zobrazuje stav operace. Výlučná sekce je zde zajištěna pomocí semaforu instancí třídy `Mutex`. Po dokončení 100% probíhajících operací se okno průběhu samo uzavře.

Kontroler mapy

Zobrazení samotného hyperspektrálního obrázku je zajištěno pomocí kontroléru mapy `ImageController`. Ten obsahuje jeden hlavní pohled `ImageCanvas` zodpovědný za vyskreslení obrázku (obrázek 5.5). Hlavní kontrolér `MainController` volá metodu `create_image` s hodnotami jednotlivých intenzit a rozměru získaného z imzML metadat. Vstupní data si `ImageCanvas` vytvoří nejprve normalizuje do barevné mapy od 0 do 255 a následně pak vytvoří instanci objektu `Fox::FXPNGImage`, u kterého každý pixel nastaví na příslušnou barvu. Pokud je měřený obrázek v jiném poměru než je velikost `ImageCanvas`, jeho zbytek je vyplněn bílou barvou. Pokud je obrázek co se počtu pixelů v jednotlivých rozměrech menší resp. větší než zobrazený `ImageCanvas`, obrázek se zvětší resp. zmenší v originálním poměru, aby nedošlo k dezinterpretaci zobrazených informací.

`ImageController` dále odchytává události vyvolané myší, konkrétně událost stisknutí levého tlačítka. Proběhne přepočet souřadnic do pozice spektra v měřeném experimentu, vykreslí se indikace zvolení konkrétního spektra a pomocí zpětného volání se informují případní zájemci o tuto událost.

Kontrolér mapy se tedy stará o zobrazení naměřených dat a případný výběr konkrétního bodu v mapě intenzity.

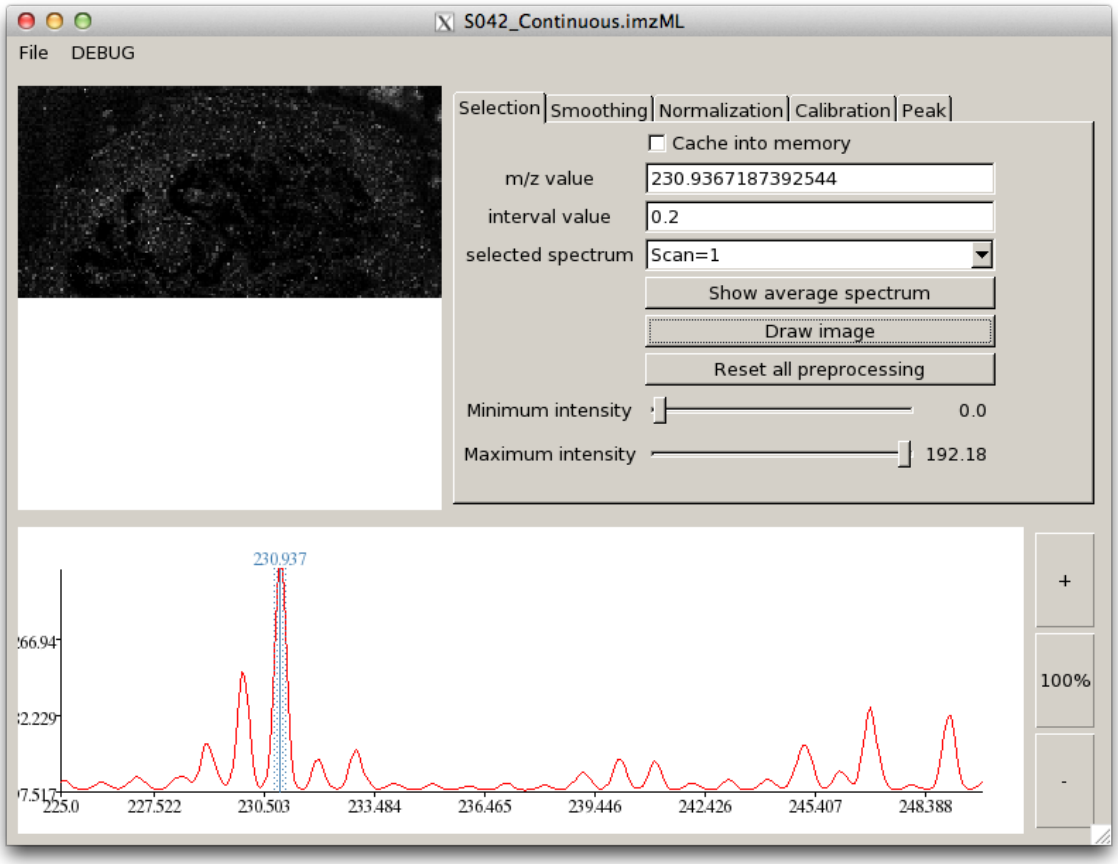
Kontrolér spektra

Dalším z klíčových kontrolérů je `SpectrumController` neboli kontrolér spektra. Ten má na starosti zobrazení dat spektra ve formě grafu a případně operace nad ním prováděné. Hlavním pohledem pro tento kontrolér je opět třída dědící z `Fox::FXCanvas` s názvem `SpectrumCanvas`. Ta má na starosti kompletní vykreslování spektra. Všechny vykreslovací operace v této třídě probíhají v metodě `draw`. Kontroléru se předávají surová spektrometrické data k zobrazení, ten si je přepočte do prostoru zabraného aktuální velikostí `SpectrumCanvas`. K přepočtu bodů do prostoru a zpět se používají metody

`spectrum_point_to_canvas` a `canvas_point_to_spectrum`. V již zmíněné metodě `draw` se při prvním výskytu dat vypočtou data, které se mají zobrazit a ty se uloží, aby se nemusely počítat znovu při dalším překreslení. Pokud dojde ke změně dimenzí pohledu či dat, zavolá se u kontroléru metoda `needs_display`, jenž označí `SpectrumCanvas` jako neplatný a nutný k překreslený v dalším vykreslovacím cyklu.

Kontrolér funguje v několika módech, které jsou určeny atributem `mode`. Pro různé procesy předzpracování jsou různé události, které lze provádět nad spektrem. V případě výběru hodnoty intenzity pro vykreslení hyperspektrálního obrázku se ve spektru vybírá intenzita a zobrazuje interval, ale při kalibraci se žádný interval nenastavuje a vybírá se více bodů.

Ve všech módech lze ve spektru pomocí myši přibližovat. Při stisku a uvolnění levého tlačítka myši dojde v aplikaci k událostem, ze kterých se v kontroléru a metod na přepočet bodu z jednotlivých prostorů vypočte do jaké oblasti se má spektrum přiblížit. Oddálení pak lze provést pomocí tlačítek umístěných vedle vykresleného spektra. Přiblížení probíhá jednoduše tak, že se vezmou originální hodnoty spektra a ořežou se do požadovaného intervalu, který se `SpectrumCanvas` předá jako nové spektrum k vykreslení.



Obrázek 5.5: Hlavní okno aplikace vytvořené v rámci projektu s ukázkou vykresleného obrázku dle vybrané intenzity

5.5 Předzpracování dat

V rámci aplikace jsou implementovány některé procesy předzpracování. Konkrétně se jedná o metody vyhlazování, kalibrace a automatického vyhledávání. V následujících sekcích je přiblížena jejich implementace v aplikaci.

Vyhlazování

Vyhlazování v aplikaci zajišťuje kontrolér vlastnosti `SmoothingFeatureController`. V rámci aplikace byly implementovány dvě metody vyhlazování a to pomocí plovoucího okna a pomocí metody Savitzky-Golay. Před samotnou aplikací lze pro obě metody zobrazit nejprve náhled jak bude vyhlazená funkce spektra vypadat při aplikování filtru s konkrétními parametry. Kontroléru jsou vždy předány hodnoty aktuálně zobrazeného spektra ve formě asociativního pole. Z tohoto pole jsou vždy vytaženy potřebné informace.

Plovoucí průměr

Nejprve bylo implementováno vyhlazování pomocí plovoucího okna. Samotné vyhlazení zajišťuje metoda `moving_average` v kontroléru `SmoothingFeatureController`. Má dva parametry, kde první je pro samotná data a druhý určuje velikost plovoucího okna. Velikost okna musí být větší než jedna a zároveň nesmí být větší než je velikost samotných dat. Vzhledem k principu jakým toto vyhlazování funguje, nedojde k vyhlazení prvních resp. posledních $n/2$ hodnot, kde n je velikost okna. Algoritmus je následující:

1. vytvoř nové pole vyfiltrovaných hodnot
2. na začátek tohoto pole přidej data z původního pole o velikosti $n/2$
3. pro každý prvek na indexu i vstupních dat v rozsahu $n/2$ až $k - n/2$, kde k je velikost vstupních dat, udělej:
   (a) vezmi prvky ze vstupního pole $i - n/2$ až $i + n/2$ a vypočti průměr
   (b) průměr ulož do pole vyfiltrovaných hodnot
   (c) pokračuj na další prvek
4. na konec vyfiltrovaných hodnot připoj $n/2$ posledních hodnot z původního pole

Savitzky-Golay

Implementace filtrování pomocí metody Savitzky-Golay je rozdělena do více metod. Nejprve se vypočtou jednotlivé váhy v metodě `weights`. Váhy jsou výsledkem řešení soustavy lineárních rovnic, která je vypočtena zjednodušenou Moore-Penrose pseudoinverzí matice. Algoritmus pak probíhá obdobně jako plovoucí průměr, jen zde hrají důležitou roli právě ony vypočtené váhy. Díky tomu výsledný algoritmus více zachovává charakteristické píky ve funkci.

Normalizace

O normalizaci se stará `NormalizationFeatureController` kontrolér. V aplikaci je možnost zvolit si mezi normalizací metodou TIC nebo normalizací za pomoci mediánu. Podobně jako u ostatních procesů předzpracování i zde má samotnou metodu na starosti blok kódu,

který se předává do hlavního kontroléru. Vstupem tohoto bloku a samotné normalizace je pak pole hodnot intenzit. Metodě `normalization` se předá takové pole společně s druhým parametrem, který určuje typ samotné normalizace. Výpočet p-normy probíhá přímo v této mětodě za pomoci funkce `inject`, zatímco výpočet mediánu je implementován jako rozšíření třídy `Array` metodou `median`.

Kalibrace

Proces kalibrace v aplikaci má na starosti kontrolér `CalibrationFeatureController`. Jeho hlavní komponenta je tabulka s kalibračními body, kterou lze vidět na obrázku 5.6 vpravo nahoře. Každý řádek této tabulky určuje právě jeden bod kalibrace. Tabulka obsahuje celkem čtyři sloupce: *selected*, *origin*, *diff* a *peptid*. První ze sloupců *selected* označuje místo v aktuálním spektru v ose intenzity. Do něj lze přímo zadat hodnotu pomocí klávesnice nebo lze bod po vybrání řádku zvolit myší přímo ve spektru. Druhý sloupec *origin* lze zadávat pouze pomocí klávesnice a měl by obsahovat hodnotu, kterou by měl vybraný bod ze sloupce *selected* obsahovat. Na základě těchto dvou hodnot pak probíhá samotná kalibrace. Třetí sloupec *diff* pouze zobrazuje rozdíl mezi prvním a druhým sloupcem a poslední sloupec *name* je pouze informativní určený pro případný název kalibrační bodu.

I u kalibrace lze před samotnou aplikací vyvolat náhled jak lze vidět na obrázku 5.6. V rámci aplikace byly implementovány dva způsoby kalibrace a to kalibrace lineární a kvadratická. Pro výpočet nové hodnoty se u obou variant používají rozdíly hodnot $origin - selected$.

Lineární kalibrace

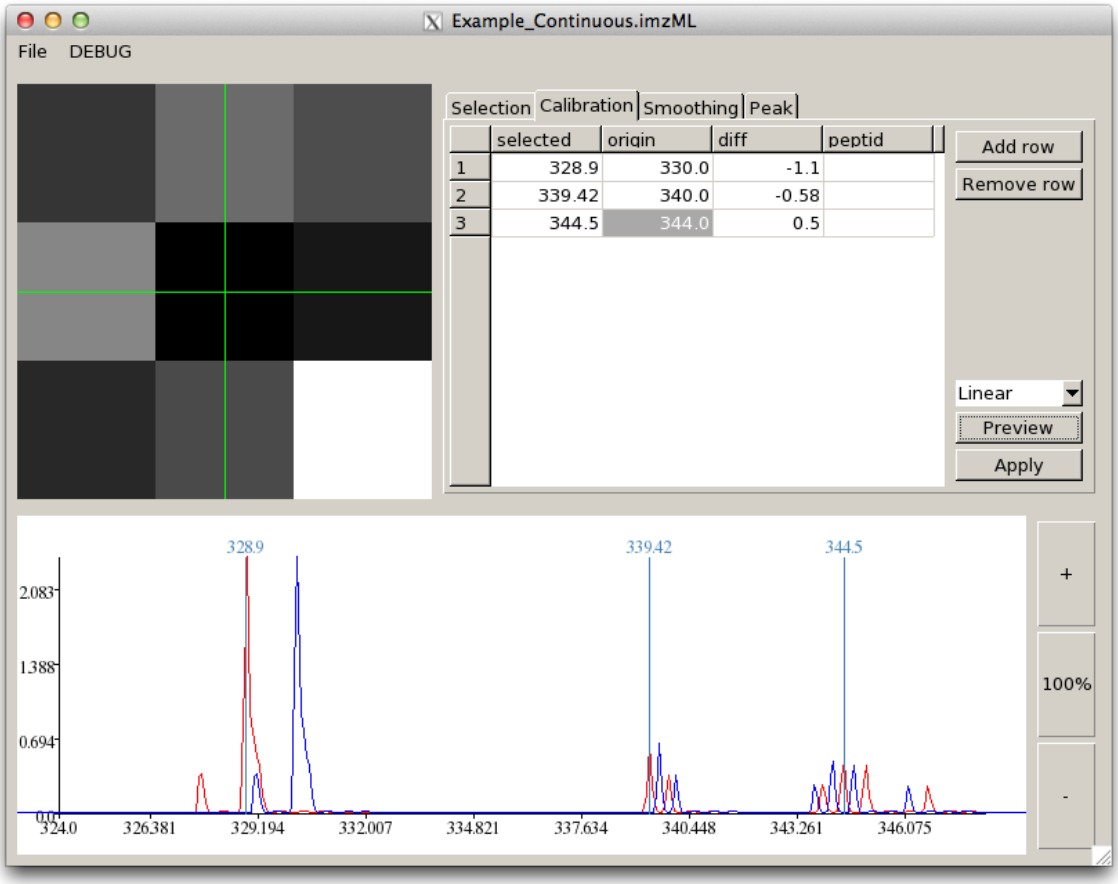
Lineární kalibrace jednoduše projde všechny body spektra a pro každý bod vypočte na základě daných kalibračních bodů jeho novou hodnotu. Tato kalibrace probíhá v metodě `linear_calibration`. Tato metoda prakticky implementuje rovnice 3.2, 3.3, které pak dosadí do rovnice 3.1 a vrátí výslednou hodnotu.

Kvadratická kalibrace

Kvadratická kalibrace je o něco složitější. Jedná se v podstatě o polynomickou aproximaci druhého řádu. Nejprve se metodou `polynomial` vypočtou koeficienty polynomu. Výpočet koeficientů je opět problém řešení soustavy rovnic. Implementace odpovídá vzorcům 3.4 resp. 3.6. Výsledkem je pole koeficientů, které se následně předá metodě `polynomial_value` společně s hodnotou funkce původního spektra x, která má být přepočtena. Nejdříve se vytvoří pole o stejné velikosti jako pole koeficientů. Do každé položky se vypočte hodnota jedné části polynomu a pomocí Ruby metody `inject` se tyto hodnoty sečtou. Na konci je ještě přičtena hodnota prvního koeficientu, který má x^0 a tudíž nefiguruje v roznásobování. Výstupem je opět kalibrovaná hodnota.

Vyhledávání píků

Vyhledávání jak bylo popsáno v kapitole 5.5 bylo čerpáno z článku [7]. V rámci tohoto článku byl algoritmus implementován a dán veřejně k dispozici ve formě zdrojového kódu v jazyce R. Konkrétně jazyk R totiž patří mezi rozšířený a používaný jazyk na implementaci různých nejen bioinformatických algoritmů včetně těch pro práci s hmotnostními spektry.



Obrázek 5.6: Náhled lineární kalibrace pomocí 3 definovaných bodů

Většina dostupných algoritmů je k dispozici na webových stránkách Bioconductor, která poskytuje nástroje pro práci s genomickými daty s otevřeným zdrojovým kódem.

RinRuby

Pro spouštění kódu v jazyce R byl použit balíček z RubyGems s názvem RinRuby. RinRuby je implementován za pomocí standardních Ruby knihoven a tudíž je knihovna jednoduše přenositelná. RinRuby jako hlavní nástroj používá TCP/IP sokety. Při každém použití RinRuby otevře systémovou rouru pro interpret jazyka R. Tato roura zůstane otevřená po celou dobu běhu Ruby skriptu, lze ji však explicitně ukončit pomocí příkazu `quit`. Všechny příkazy jazyka R se zadávají pomocí metody `eval`, která tyto příkazy pošle přes onu otevřenou rouru. Dá se říct, že Ruby představuje server a R klient, kteří mezi sebou komunikují skrze rouru. Přes tuto rouru je možné si předávat data z R skriptu do Ruby skriptu či naopak. Jelikož k tomu dochází přes sokety, předávání velkých objemů dat je rychlé. [4]. Ukázka použití RinRuby v Ruby skriptu:

```
require 'rinruby'

# několik způsobů přiřazení hodnoty do interpretu R
R.a = 1000
```

```
R.b = 5
R.assign(,,c‘‘, 3)

# vykonání kódu R
R.eval <<EOF
  x <- rgamma(a, b, c)
  hist(x)
EOF

# výpis hodnoty z interpretu R v Ruby
puts R.x
```

V aplikaci je použit balíček z Bioconductoru s názvem *MassSpecWavelet*. Balíčku je z uživatelského rozhraní předán parametr, určující počet škál použitý v algoritmu hledání. Dále následuje průběh samotného kódu v jazyce R:

```
# načítání balíčků
library(MassSpecWavelet)
library(waveslim)

# vyhledávací algoritmus
peakInfo <- peakDetectionCWT(exampleMS)

# uložení potřebných výsledků do proměnných
majorPeakInfo <- peakInfo$majorPeakInfo
peakSNR <- majorPeakInfo$peakSNR
allPeakIndex <- majorPeakInfo$allPeakIndex
```

První dva řádky načtou do prostředí jazyka R balíček *MassSpecWavelet* se samotným algoritmem hledání a balíček *waveslim* obsahující pomocné funkce pro práci se signály. Na dalším řádku je volaná funkce `peakDetectionCWT`, která obaluje základní komponenty samotného algoritmu a provádí celý výpočet. V této funkci se volají funkce pro výpočet spojité vlnkové transformace `cwt`, funkce hledající lokální maxima `getLocalMaximumCWT`. Hřbety v koeficientech spojité vlnkové transformace hledá funkce s názvem `getRidge`. Poslední důležitou část zajišťuje funkce `identifyMajorPeaks`, která identifikuje píky na základě hřbetů z 2D matice koeficientů vlnkové transformace a odhadnutým poměrem signálu k šumu. Poslední tři řádky pak uloží důležité informace z výsledku hledání. Pro proběhnutí R skriptu se v Ruby ještě nalezené výsledky vyfiltrují, aby obsahovaly pouze píky, které mají poměr signálu k šumu větší nebo roven zadané hodnotě z uživatelského rozhraní.

Fronta předzpracování

Jednotlivé kroky předzpracování lze nejen prohlížet, ale také aplikovat na prohlížená data. Aplikace má za tímto účelem vytvořenou frontu úloh, která se provádí před každým přístupem k datum z měření. Tato fronta je uložena v hlavním kontroléru `MainController` v atributu `preprocess`. Tento atribut obsahuje asociativní pole `Hash`, kde každá operace předzpracování má svého zástupce. Například vyhlazování lze nalézt pod klíčem `:smoothing`. Každý takový krok předzpracování obsahuje instanci objektu procesu `Proc` neboli bloku kódu. Při přístupu k binárním datům se kontrolér podívá, zda v tomto atributu neexistují

kroky pro předzpracování a pokud ano, tak je provede. V uživatelském rozhraní se pak pracuje s daty takto upravenými.

Kroky předzpracování vznikají ve svých kontrolérech vlastností. Zde proběhne vytvoření samotného objektu `Proc` a s právě dvěma parametry: pole intenzit a pole hodnot m/z. Proces s těmito dvěma parametry byl takto vytvořen, protože každý krok předzpracování může pracovat s jinou částí dat, například vyhlazování zajímají pouze hodnoty intenzit, zatímco kalibrace upravuje hodnoty m/z. V procesu pak dochází k příslušné operaci předzpracování. Tento objekt procesu se v rámci jednotlivých kontrolérů vlastností používá i pro náhled před aplikací předzpracováním.

Každý typ kroku předzpracování může být ve frontě právě jednou. Z úvodní záložky výběru, lze pak tyto kroky vymazat. Kroky předzpracování probíhají pouze za běhu aplikace.

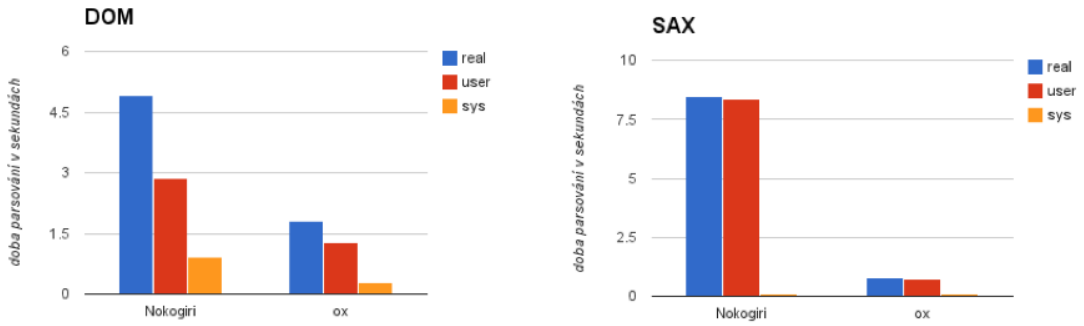
Kapitola 6

Experimenty s daty

V následující kapitole budou provedeny experimenty se vzorovými daty na aplikaci vytvořené v rámci této práce. Byly použity jak data ukázková, tak data z reálného měření z laboratoře. Všechny experimenty byly provedeny na CPU Intel Core™ i5-3317U CPU s taktovací frekvencí 1.70GHz s 4GB operační paměti na systému Mac OS X.

6.1 Čtení dat

Jako součást aplikace vznikl i balíček pro čtení nového standardu imzML. Jak bylo uvedeno v kapitole implementace, ke čtení byl použit SAX parser *Ox*. Díky tomu čtení imzML souboru dosahuje mnohem lepších výsledků než v případě použití knihovny *Nokogiri*. Jako test rychlosti čtení posloužil reálný imzML soubor s názvem `S043_Processed.imzML` o velikosti 43.1MB (obrázek 6.1). Soubor byl otevřen v daném módu a přečten bez ohledu na sémantiku obsahu.

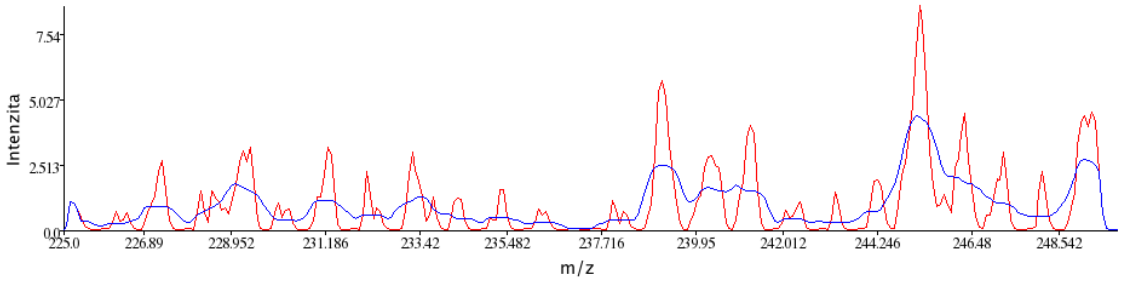


Obrázek 6.1: Výsledky testu parsování XML knihoven v Ruby

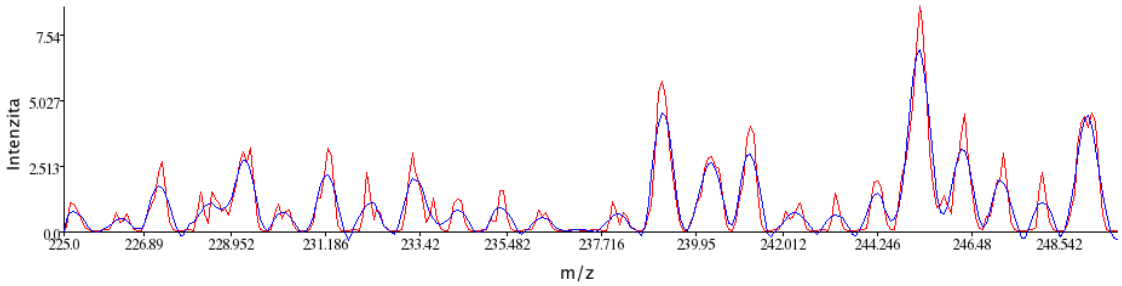
Rychlost čtení je důležitá, u stávajícího software je běžné, že například komerční aplikace flexImaging otevírá jeden běžný experiment zobrazovací hmotnostní spektrometrie 15 min. Porovnávat však tento čas s časem otevírání v aplikaci vytvořené v rámci práce není možné neboť komerční aplikace data načítá pravděpodobně do nějaké vlastní databáze či formátu, ze které pak přistupuje k datům rychleji. Zároveň tento komerční software čte data ze spousty malých souborů, zatímco vytvořená aplikace tyto data čte z jednoho binárního souboru.

6.2 Vyhlazování dat

Účinnost implementovaných vyhlazovacích metod lze díky náhledu aplikace konkrétního filtru přímo v aplikace snadno pozorovat. Na obrázcích 6.2 a 6.3 lze vidět rozdíly vyhlazování pomocí plovoucího okna a Savitzky-Golay. První uvedený filtr s rostoucí šířkou více rozšiřuje a zmenšuje jejich amplitudu. Píky umístěné příliš blízko u sebe dokonce zruší. U tohoto filtru je velmi důležité rozlišení naměřených dat a zvolená velikost okna. Při špatně zvolených takovýchto hodnot dochází k velkému poškození dat a tím ke ztrátě informace. Filtr Savitzky-Golay vykazuj mnohem lepší výsledky při zachování jednotlivých píku. Při jeho použití velikost píku tolik nezmenšuje a zachovává malé píky i při stejně velkém filtrovací okně.



Obrázek 6.2: Náhled vyhlazení spektra pomocí algoritmu plovoucího průměru s oknem o velikosti 11

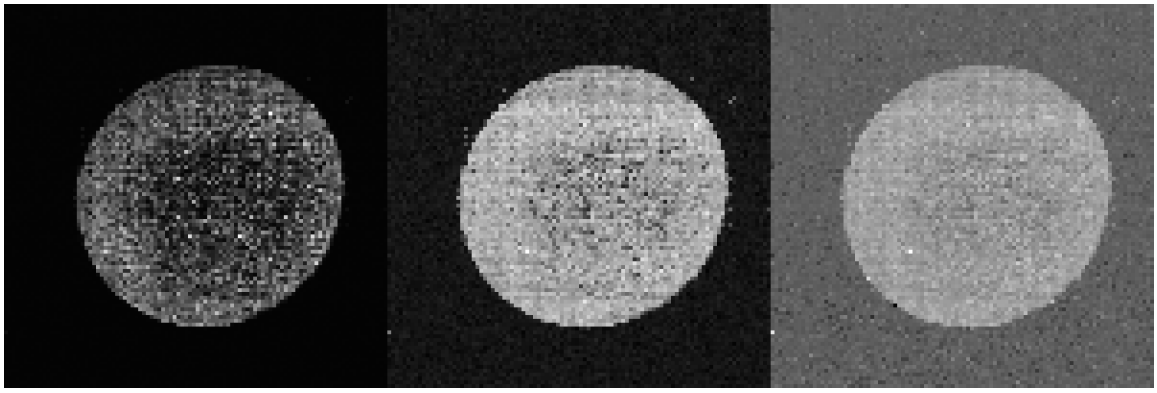


Obrázek 6.3: Náhled vyhlazení spektra pomocí algoritmu Savitzky-Golay s oknem o velikosti 11

6.3 Normalizace

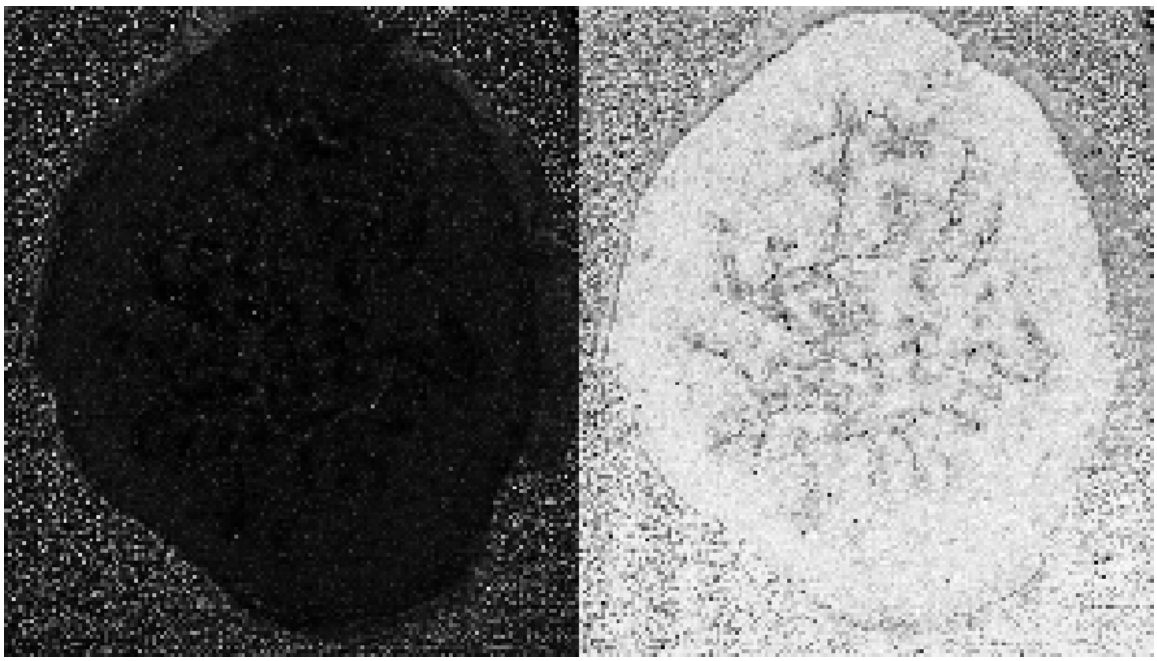
Hlavním cílem normalizace je sjednotit škály intenzit napříč celým obrázek zobrazovací hmotnostní spektrometrie. Hyperspektrální obrázek zobrazuje právě intenzity a jejich sjednocení by mělo vést k lepším výsledkům zobrazení. Pro experiment byly zvoleny dva experimenty. První z nich obsahuje *calmix* neboli kalibrační směs ze souboru `20121220_LIN_100x100_1mmScan_PAPER_0018_spot5_1855`. Na měřící destičce bylo nanesen vzorek ve formě kolečka (obrázek 6.4).

Pro druhý experiment byl použit obrázek řezu močového měchýře (obrázek 6.5). Lze jde vidět výrazné zlepšení vizualizovaných dat. Na normalizovaném obrázku lze oproti pů-



Obrázek 6.4: Výsledek aplikace normalizace. Na obrázku vlevo je původní naměřený obrázek bez jakékoliv úpravy. Uprostřed lze vidět ten samý obrázek s aplikovanou normalizací TIC. Obrázek vpravo pak zobrazuje výsledek normalizace pomocí mediánu.

vodnímu pozorovat oblasti vnitřku močového měchýře, které v původním zobrazení nejsou téměř vidět.



Obrázek 6.5: Normalizace na obrázku močového měchýře, vlevo originální obrázek a vpravo obrázek s normalizovanými daty pomocí TIC.

Kapitola 7

Závěr

V průběhu práce vznikla aplikace pro čtení a vizualizaci dat zobrazovací hmotnostní spektrometrie pro unikátní, nekomerční přístroj. Díky možnosti přístupu do laboratoře se sestaveným hmotnostním spektrometrem z vlastních prostředků jsem spolupracoval s vědci, kteří se na sestavení přístroje podíleli, a software přizpůsobil jejich požadavkům.

V rámci práce jsem se seznámil se základními principy hmotnostní spektrometrie (kapitola 2), zvláště pak s typem MALDI TOF. Většina práce se pak věnuje zobrazovací hmotnostní spektrometrií. Byly prozkoumány způsoby ukládání dat z hmotnostních spektrometrů, jejich reprezentace a možnosti jejich předzpracování (kapitola 3). V práci je stručně shrnut (kapitola 4) současný stav existujícího software pro zpracování a vizualizaci dat hmotnostní spektrometrie a na základě požadavků je navržena a implementována nová aplikace pro zpracování a vizualizaci (kapitola 5).

Aplikace, která vznikla v rámci této práce umí pracovat jako jedna z mála s novým standardem imzML. Pro čtení tohoto standardu vznikl samostatný balíček s názvem imzml v jazyce Ruby. Samotná vizualizační aplikace tento balíček využívá, ale je možné jej použít i zvlášť pro jiné účely. Díky tomuto balíčku je možné otevírat i běžné větší soubory z experimentů, které v jinde otevřít nelze. Oproti existujícímu software má výhodu v tom, že umí data nejen přečíst a zobrazit, ale umí na data také aplikovat základní procesy předzpracování. Zdrojový kód je veřejně dostupný. Práce je převážně psána ve vysoko úrovňovém objektově orientovaném jazyce Ruby a lze ji snadno přizpůsobovat konkrétním experimentům či potřebám vědců v laboratoři. Na toto bylo myšleno již při návrhu, který je pro takové budoucí úpravy připraven. Příkladem může být ukázkové rozšíření funkčnosti pomocí skriptu v jazyce R pro vyhledávání píků.

V současné době je aplikace funkční a ozkoušena na datech (kapitola 6) z konkrétního přístroje partnerské laboratoře, se kterou bude spolupráce pokračovat i nadále. Pro širší využití je nutné na aplikaci ještě zapracovat - otestovat na datech pořízených jinými přístroji v jiných laboratořích. Na komerčních aplikacích, které se v současné době používají, pracují celé týmy lidí a ceny takových aplikací jsou velmi vysoké. Vytvoření veřejně otevřené verze dává možnost nejen využití, ale i případné spolupráce více lidí.

V budoucnosti by bylo možné některé části přepsat do jazyka C, díky čemuž by se některé výpočty zrychlily. Toto však může vést k omezení přenositelnosti na jiné platformy. Každému kroku předzpracování se lze věnovat samostatně a je možné tyto části aplikace vylepšit či doplnit o nové přístupy. Aplikace by také v budoucnu mohla být rozšířena o možnost pokročilejší analýzy dat jako například automatické vyhledávání oblastí. Díky stále větším možnostem nejen počítačů, ale i samotných přístrojů, je tato oblast předmětem intenzivního výzkumu při hledání biomarkerů či studiu chemického složení biosystémů.

Literatura

[1] Alexandrov, T.: MALDI imaging mass spectrometry: statistical data analysis and current computational challenges. *BMC Bioinformatics*, 2012.

[2] Apple Inc.: Concepts in Objective-C Programming. January 2012.
URL https://developer.apple.com/library/ios/documentation/general/conceptual/CocoaEncyclopedia/CocoaEncyclopedia.pdf

[3] Bednařík, A.; Kuba, P.; Moskovets, E.; aj.: Rapid Matrix-Assisted Laser Desorption/Ionization Time-of-Flight Mass Spectrometry Imaging with Scanning Desorption Laser Beam. *Anal. Chem.*, 2014, ISSN 0003-2700, doi:http://dx.doi.org/10.1021/ac402823n.

[4] Dahl, D. B.; Crawford, S.: RinRuby: Accessing the R Interpreter from Pure Ruby. *Journal of Statistical Software*, 11 2009, ISSN 1548-7660.
URL http://www.jstatsoft.org/v29/i04

[5] Daubechies, I.: *Ten Lectures on Wavelets*. SIAM: Society for Industrial and Applied Mathematics, May 1992, iSBN 978-0898712742.

[6] Deininger, S.-O.; Cornett, D. S.; Paape, R.; aj.: Normalization in MALDI-TOF imaging datasets of proteins: practical considerations. *Anal. Bioanal. Chem.*, April 2011.

[7] Du, P.; Kibbe, W. A.; Lin, S. M.: Improved peak detection in mass spectrum by incorporating continous wavelet transform-based pattern matching. *Bioinformatics*, July 2006.

[8] Fenn, J. B.; Mann, M.; Meng, C. K.; aj.: Electrospray ionization for mass spectrometry of large biomolecules. *Science*, 2007.

[9] Johnson, L.: *FXRuby - Create Lean and Mean GUIs with Ruby*. Pragmatic Prammers, LLC, March 2008, iSBN 1-934356-07-7.

[10] Jonsson, F.: SGFilter - A stand-alone implementation of the Savitzky–Golay smoothing filter. December 2011.
URL http://jonsson.eu/programs/cweb/sgfilter/

[11] Karas, M.; Hilenkamp, F.: Laser desorbtion ionization of proteins with molecular masses exceeding 10,000 daltons. *Analitical Chemistry*, 1988.

[12] Klinkert, I.; Chughtai, K.; Ellis, S. R.; aj.: Methods for full resolution data exploration and visualization for large 2D and 3D mass spectrometry imaging datasets. *International Journal of Mass Spectrometry*, December 2013.

[13] Kuba, P.: Zpracování a vizualizace dat z hmotnostního spektrometru typu TOF-MALDI. *Vysoké Učení Technické v Brně*, 2012.

[14] Mamyrin, B. A.; Karataev, V. I.; Shmikk, D. V.; aj.: The mass-reflectron, a new nonmagnetic time-of-flight mass spectrometer with high resolution. *Sov. Phys.*, 1973.

[15] Martens, L.; Chambers, M.; Sturm, M.; aj.: mzML - a Community Standard for Mass Spectrometry Data. Jan 2011.
URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3013463/

[16] Perrotta, P.: *Metaprogramming Ruby: Program Like the Ruby Pros*. The Programatic Programmers, 2013, iSBN 978-1-93435-647-0.

[17] Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; aj.: *Numerical Recipes in C - The Art of Scientific Computing Second Edition*. Cambridge University Press, 1992, iSBN 0-521-43108-5.

[18] Robichaud, G.; Garrard, K.; Barry, J.; aj.: MSiReader: An Open-Source Interface to View and Analyze High Resolving Power MS Imaging Files on Matlab Platform. *Journal of the American Society for Mass Spectrometry*, March 2013.

[19] Savitzky, A.; Golay, M.: Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Anal. Chem.*, July 1964.

[20] Schramma, T.; Hestera, A.; Klinkertb, I.; aj.: imzML — A common data format for the flexible exchange and processing of mass spectrometry imaging data. *Journal of Proteomics*, 2012.

[21] Stephens, W.: A Pulsed Mass Spectromemeter with Time Dispersion. *Phys. Rev.*, 1946.
URL http://link.aps.org/abstract/PR/v69/p674/s2

[22] Stoeckli, M.: BioMap.
URL http://www.maldi-msi.org/index.php?option=com_content&view=article&id=14&Itemid=32

[23] Stoeckli, M.: MSImageViewer.
URL http://www.maldi-msi.org/index.php?option=com_content&view=article&id=315&Itemid=102

[24] Strohalm, M.; Kavan, D.; Novák, P.; aj.: mMass 3: A Cross-Platform Software Environment for Precise Analysis of Mass Spectrometric Data. *Anal. Chem 82*, 2010.

[25] Tanaka, K.; Waki, H.; Ido, Y.; aj.: Protein and Polymer Analyses up to m/z 100 000 by Laser Ionization Time-of flight Mass Spectrometry. *Rapid Commun Mass Spectrom*, 1988.

[26] Thomas, D.; Fowler, C.; Hunt, A.: *Programming Ruby 1.9 & 2.0*. The Pragmatic Programmers, 2013, iSBN 978-1-93778-549-9.

[27] Thomson, J.: Rays of positive electricity. *Proceedings of the Royal Society*, 1913.

[28] Wikipedia: Lineární regrese. 2014, [Online; ze dne 8.5.2014].
URL http://cs.wikipedia.org/wiki/Lineárn%C3%AD_regrese

[29] Wikipedia: Mexican Hat Wavelet. 2014, [Online; ze dne 8.5.2014].
URL http://en.wikipedia.org/wiki/File:MexicanHatMathematica.svg

[30] Wikipedia: Polynomická regrese. 2014, [Online; ze dne 8.5.2014].
URL http://cs.wikipedia.org/wiki/Polynomická_regrese

[31] Xiang, B.; Prado, M.: An Accurate and Clean Calibration Method for MALDI-MS. *Journal of Biomolecular Techniques*, 2010.

Dodatek A

Obsah CD

Na přiloženém CD jsou zdrojové soubory práce ve formátu LaTeXvčetně všech použitých obrázků. Tyto soubory se nachází ve složce `tex`.

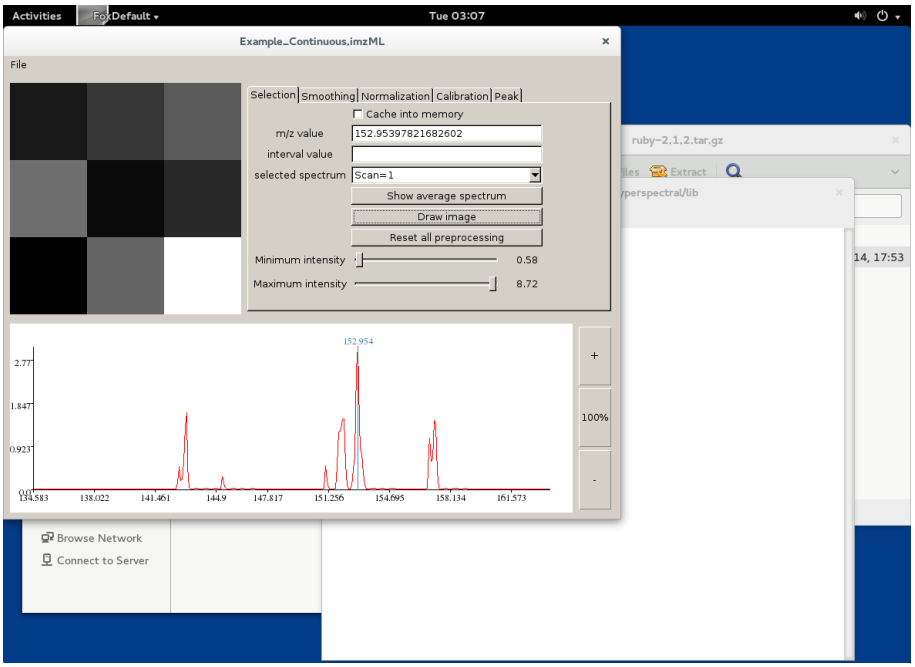
Ve složce `source` lze pak nalézt jednotlivé zdrojové kódy, které vznikly v průběhu práce. Složka `hyperspectral` obsahuje zdrojové kódy samotné aplikace, složka `imzml` pak zdrojové kódy k balíčku pro čtení imzML dat. Dále jsou přiloženy zdrojové kódy, za pomocí kterých byla testována rychlost XML parserů v Ruby.

V neposlední řade jsou na CD ukázkové soubory imzML a to jak vzorové poskytované společně se standardem tak některé z reálných experimentů z laboratoře. Tyto soubory lze nalézt v adresáři `imzml`. Dále jsou k dispozici OBO soubory s definicemi v adresáři `obo`.

Dodatek B

Návod

V této části je stručně popsán jak aplikaci zprovoznit a to na systému OS X od společnosti Apple. Aplikaci lze spustit i na jiných systémech, konkrétně je zde uveden postup jak aplikaci spustit na systému Linux Fedora (obrázek B.1), ale aplikace na tomto systému nebyla řádně otestována.



Obrázek B.1: Aplikace na systému Linux Fedora

B.1 Instalace

Aplikace byla vyvíjena a testována na systému OS X 10.9.2. Pro její běh je nutné mít v systému nainstalované a zprovozněné, některé nástroje a knihovny. Prvně je nutné nainstalovat Ruby interpret a to minimálně ve verzi 2.1. Pokud je již na systému nainstalována jiná verze Ruby, lze toho jednoduše dosáhnout pomocí nástrojů `rbenv` nebo `rvm`. Alternativou může být instalace z balíčku na konkrétním systému (pro OS X například `homebrew` či `macports`, pro Linux Fedora nástroj `yum`) či zkompilování přímo ze zdrojových kódu. Po instalaci je

nutné pomocí nástroje `gem` nainstalovat balíček s názvem `bundler`. Jeho následné použití bude dále popsáno.

Dalším nutným krokem je instalace knihovny FOX Toolkit. Tu lze taktéž nainstalovat pomocí již zmíněných nástrojů či přímo ze zdrojových kódů. Aplikace byla vyvíjena nad stabilní verzí 1.6.49. Pro verzi 1.7.x či vyšší není aplikace přispůsobena, doporučuji použít verzi z větve 1.6.x.

Dále je nutné mít v systému prostředí X11, které vyžaduje knihovna FOX Toolkit pro svůj běh. Systém Linux většinou tuto závislost obsahuje, na OS X je nutné doinstalovat XQuartz.

Poslední externí závislostí je interpret jazyka R. Ten je nutné pro správnou funkci automatického vyhledávání vrcholků a lze opět nainstalovat pomocí již zmíněných nástrojů. Po nainstalování je ještě nutné přímo v interpretu stáhnout a nainstalovat balíček z Bioconductory s názvem `MassSpecWavelet`.

B.2 Spuštění a ovládání

Před prvním spuštěním je nutné doinstalovat ještě závislosti v Ruby. Toho lze jednoduše dosáhnout pomocí příkazu `bundle install` ve složce `hypespectral/lib`. Ze souboru `Gemfile` se zjistí a nainstalují potřebné závislosti. Poté již pro samotné spuštění stačí ve složce `lib` spustit příkaz

```
ruby hyperspetral.rb
```

B.2.1 Načtení souboru a zobrazení hyperspektrálního obrázku

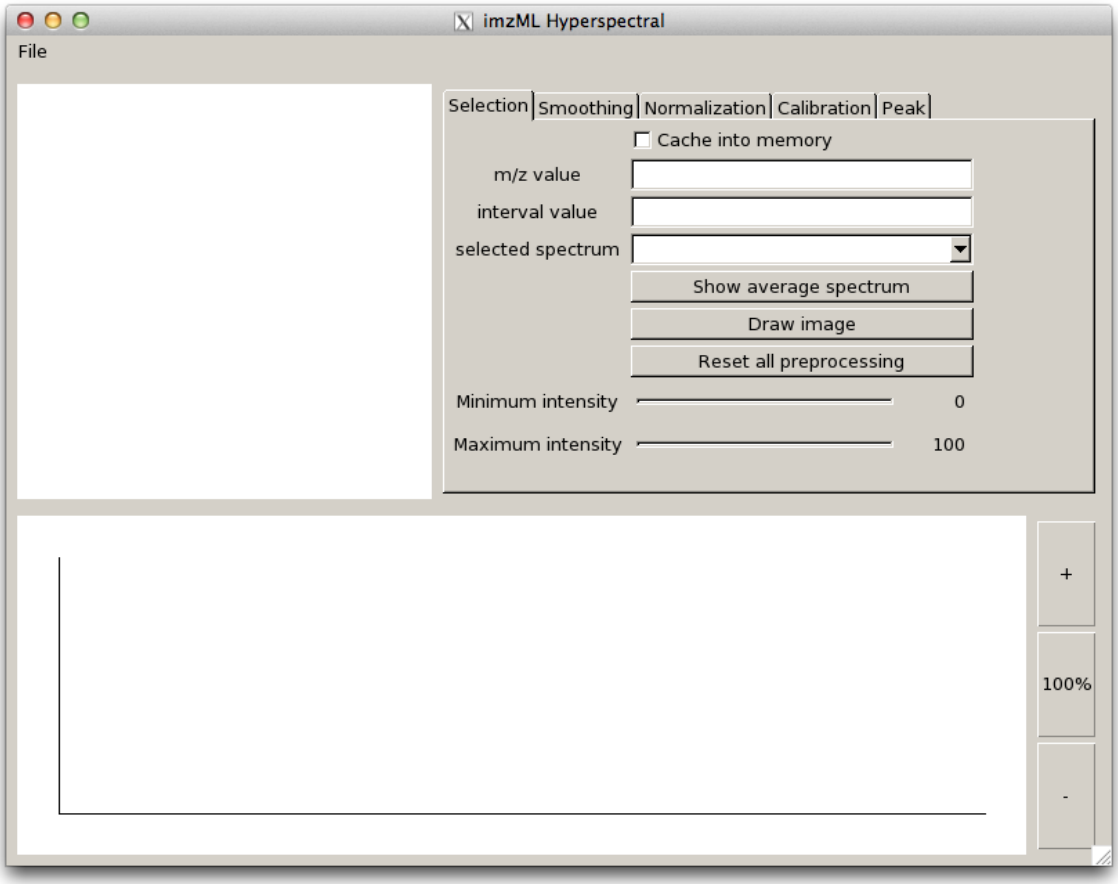
Po otevření hlavního okna v menu výběrem `File > Open` vybereme zdrojový imzML soubor. Aplikace předpokládá, že se binární soubor ibd jsmenuje stejně a je umístěn ve stejné složce. Po načtení metadat se v záložce *Selectiion* vyplní měřená spektra a v dolní části se zobrazí první spektrum z měření jak lze vidět na obrázku B.2

Pro zobrazení hyperspektrálního obrázku je nutné prvně zvolit hodnotu *m/z* a to buď pomocí kliknutí pravé myši ve spektru nebo zadáním hodnoty do pole *m/z*. Pro přiblížení ve spektru na konkrétní oblast lze použít buď tlačítka v pravo od spektra neb pravým klikem myší a následným tažení na oblast zájmu. K hodnotě *m/z* lze také vybrat interval, který se bude brát v úvahu při generování obrázku. V části spektra lze zobrazit jakékoliv konkrétní spektrum z měření a nebo průměrné spektrum napříč celým experimentem pomocí tlačítka *Show average spectrum*. I v takovém spektru lze vybírat bod pro zobrazení obrázku, je to jen jiný úhel pohledu. Spektra je možné také měnit skrze obrázek a to kliknutím na konkrétní místo v obrázku. U zobrazeného obrázku lze omezit intenzity, které jsou vykreslené pomocí dvou posuvníků. Pokud je zatrhnuté políčko *Cache into memory* tak se při následujícím načítání hodnoty uloží do paměti pro rychlejší přístup. Tato operace může být ale pamětově velmi náročná v závislosti na velikosti vstupních dat.

Po vybrání hodnoty *m/z* lze vygenerovat obrázek pomocí tlačítka *Draw image* a výsledek může vypadat jako na obrázku B.3

B.2.2 Předzpracování

Data lze v aplikaci předzpracovat několika způsoby. Z každé kategorie lze na data aplikovat pouze jeden krok předzpracování, všechny tyto kroky jde pak na záložce *Selection* vynulo-

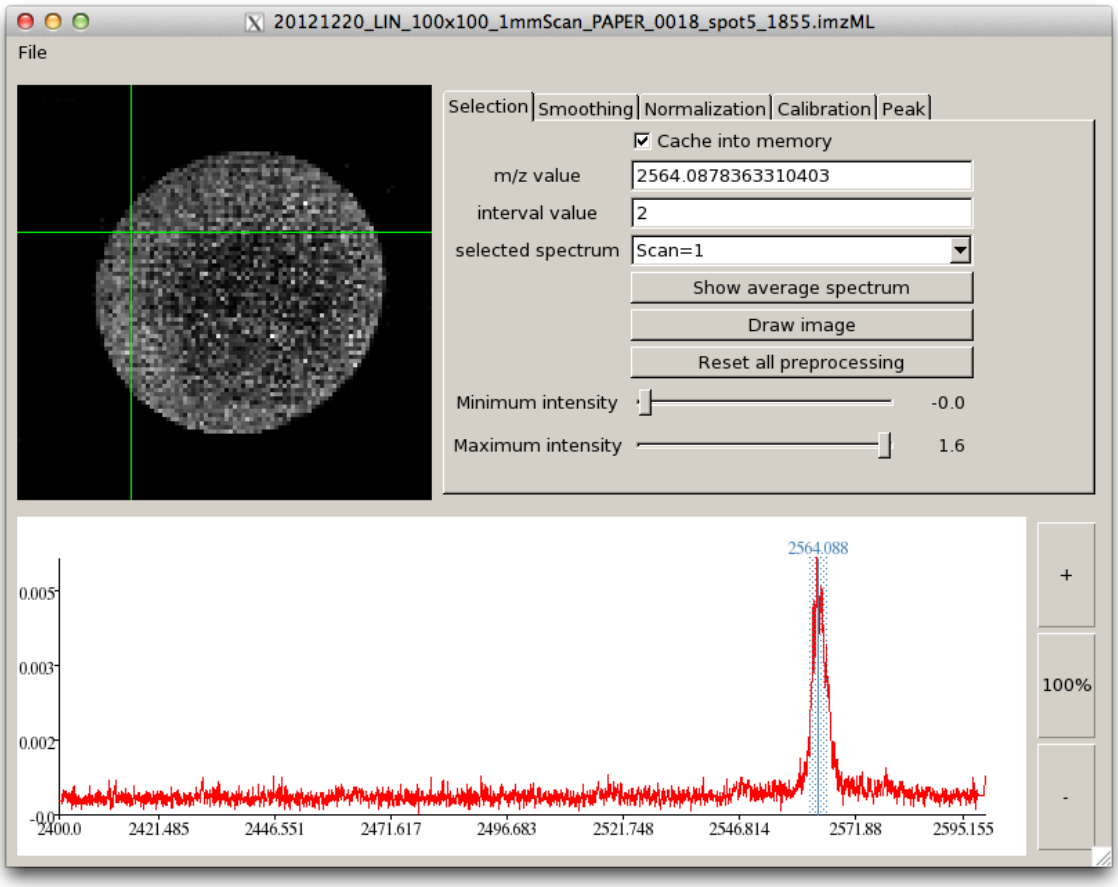


Obrázek B.2: Stav aplikace po otevření souboru

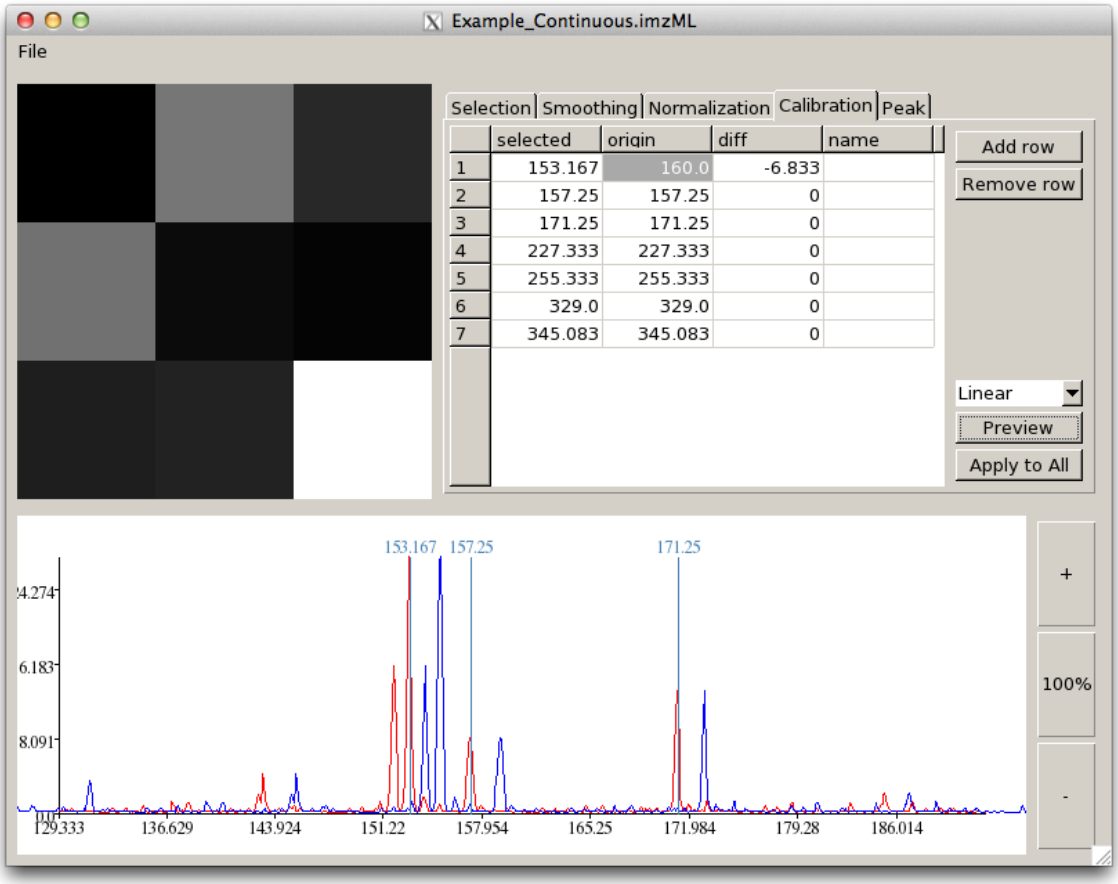
vat pomocí tlačítka *Reset all preprocessing*. Důsledky některých operací lze před aplikací prohlédnout pomocí *Preview* jak je vidět na obrázcích 6.2 či 6.3.

Pomocí automatického vyhledávání na záložce *Peak* lze nalézt některé vrcholky automaticky. Nalezených vrcholků může být více a jejich hledání lze omezit zadáním hodnoty poměru šumu k signálu, která je hraniční do položky *SNR*. Výsledek lze pak naimportovat do sekce kalibrace, kde zadávání vrcholků skrze tabulku může být zdlouhavé.

Pokud se aplikuje nějaký krok předzpracování, je nutné vždy znovu zmáčknout tlačítko *Draw image* pro aplikaci všech zvolených kroků.



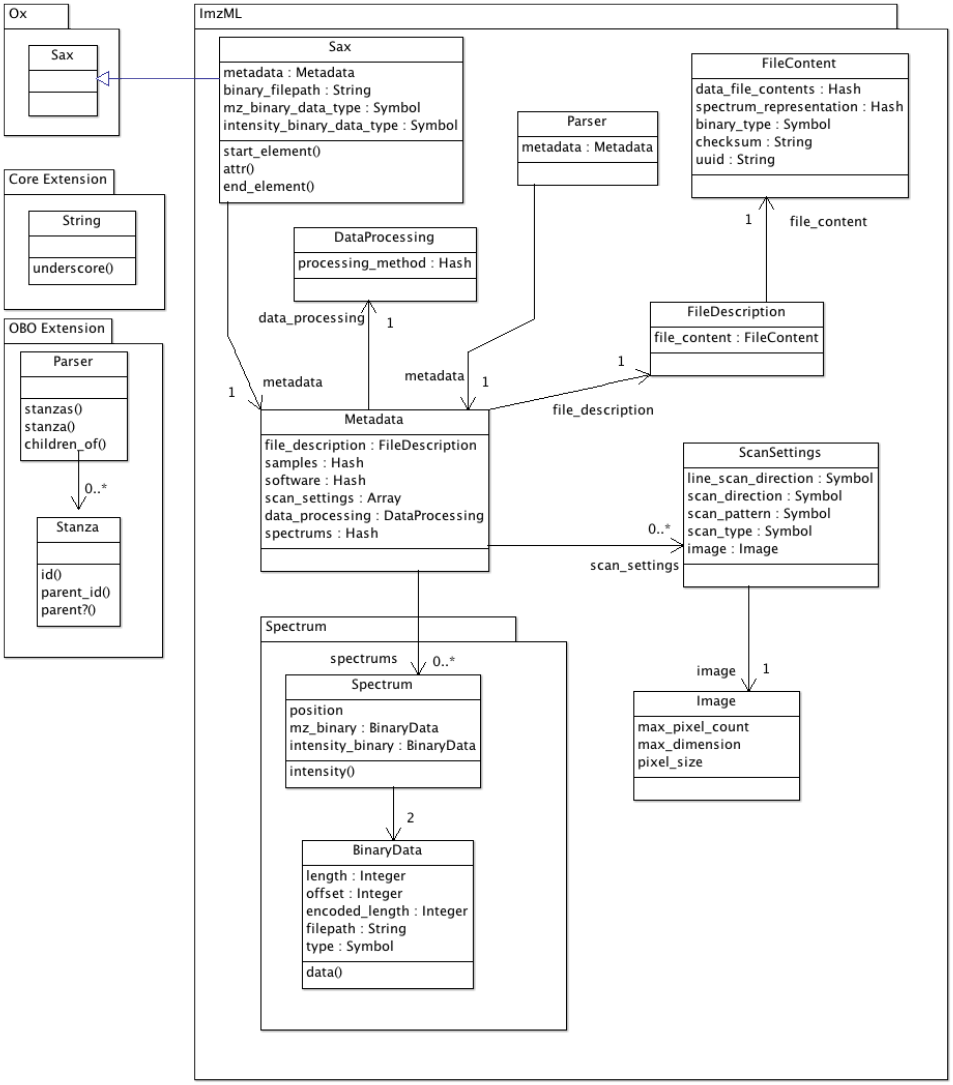
Obrázek B.3: Zobrazení hyperspektrálního obrázku



Obrázek B.4: Ukázka náhledu kalibrace pomocí naimportovaných vrcholků z automatického vyhledávání píků

Dodatek C

Třídní diagram balíčku imzml



Obrázek C.1: