Aus dem Institut für Medizinische Informatik
der Hochschule Mannheim

A Software Ecosystem for Remote Analysis of Mass Spectrometry Imaging Data

Inauguraldissertation
zur Erlangung des Doctor scientiarum humanarum (Dr. sc. hum.)
der
Medizinischen Fakultät Mannheim
der Ruprecht-Karls-Universität
zu
Heidelberg

vorgelegt von

Jonas Tom Frederik Cordes, M. Sc.

aus
Heidelberg

2024

Dean: Prof. Dr. med. Sergij Goerdt
Referent: Prof. Dr. sc. hum. Ivo Wolf

Abstract

A Software Ecosystem for Remote Analysis of Mass Spectrometry Imaging Data

In many areas of biomedical research, images are crucial for scientific progress. Interactive access to these images is essential, enhancing understanding and facilitating advancements, particularly in fields like pathology, radiology, and cellular biology. As imaging techniques continue to advance, generating ever more detailed datasets, the amount of data to be stored and processed will continue to grow. Consequently, data and computationally intensive processes are being increasingly relocated to centralized resources with substantial storage and processing capabilities. However, large, multidimensional and multi-modal biomedical images, such as those generated in experiments with mass spectrometry imaging, pose a major challenge for fast, comprehensive and interactive remote access. Processes as image data exploration, image analysis, the development of new image analysis methods, and interdisciplinary collaboration of domain experts can be hampered if data-intensive transfers to local systems are required, e.g. for processing of images with interactive applications domain experts are familiar with. Current efforts to utilize remote resources focus on providing integrated environments for remote development and applications for execution of reproducible image analysis, while lacking comprehensive interactive capabilities to work with high-dimensional image data.

The first part of this work introduces advanced interactive access strategies for multi-modal 2D/3D mass spectrometry imaging (MSI) datasets. Concepts for fast, memory-efficient interactive access to imzML mass spectrometry imaging datasets are presented, which facilitate advanced interactive workflows such as multi-modal image fusion and 3D image reconstruction. The effectiveness of the concepts are demonstrated within the context of the novel and openly accessible desktop application called *Mass spectrometry imaging Applications for Interactive Analysis in MITK* (M²aia). Furthermore, concepts for a programming language-independent integration of third-party command-line applications via Docker (mitk-docker) into the interactive framework of M²aia are presented. Finally, concepts for an optimized MSI data access for deep learning are proposed and shown in combination with the data handling and processing capabilities of M²aia as part of a python package (pyM²aia).

The second part of this thesis proposes a versatile and efficient interactive remote working environment. It relies on interactive containerized applications that can be deployed with Docker and accessed using a web browser. The effectiveness of the concept is demonstrated by applying it to a diverse set of biomedical image processing applications, M²aia for MSI data, MITK for clinical images, ImageJ for microscopy images, QuPath for manual segmentation of histology images, and ilastik for semi-automatic segmentation of a wide range of biomedical imaging modalities. Access to these remote-controlled applications facilitates a variety of interactive tasks on remote image data such as image analysis, method development and collaboration with experts.

In both parts of this work, diverse use-cases are elaborated to show the capabilities of the respective concepts. Use-cases demonstrate the advanced interactive capabilities of M²aia with respect to multi-modal image fusion and 3D image reconstructions. A comprehensive set of MSI-based deep learning use-cases is realized to showcase the data access capabilities of pyM²aia. Furthermore, the seamless integration of Docker-based applications into the interactive environment of M²aia is demonstrated. Finally, capabilities of the interactive remote working environment are demonstrated.

In summary, this thesis introduces comprehensive concepts for processing and interactive analysis of multi-modal 2D/3D mass spectrometry image data as well as additional concepts for a general support of interactive remote working applying to a wide range of interactive biomedical image processing tasks.

Contents

Acronyms

2D Two-Dimensional

3D Three-Dimensional

API Application Programming Interface

CBCT Cone-Beam Computed Tomography

CSV Comma-Separated Values

CT Computed Tomography

DESI Desorption Electrospray Ionization

DKFZ German Cancer Research Center

DL Deep Learning

FFPE Formalin Fixation Paraffine Embedded

FTICR Fourier-transform ion cyclotron resonance

FTIR Fourier-Transform Infrared Spectroscopy

GUI Graphical User-Interface

hws half-window size

H&E Hematoxylin and Eosin

IDE Integrated Development Environments

IF ImmunoFluorescence Microscopy

IHC Immunohistochemistry

IMC Imaging Mass Cytometry

MALDI Matrix-Assisted Laser Desorption/Ionization

MI Mutual Information

MITK Medical Imaging Interaction Toolkit

MPM Molecular Probabilistic Map

MPR Multiplanar Reconstruction

MRI Magnetic Resonance Imaging

MRP Mass Resolving Power

MS Mass Spectrometry

MSI Mass Spectrometry Imaging

NCC Normalized Cross-Correlation

OSGi Open Services Gateway initiative

PCA Principal Component Analysis

RMS root mean square

RWE Remote Working Environment

SIMS Secondary Ion Mass Spectrometry

SSD Sum of Squared Differences

t-SNE t-Distributed Stochastic Neighbor Embedding

TIC total-ion-count

TOF Time-of-Flight

TRE Target Registration Error

UI User-Interface

VNC Video Network Computing

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Motivation

In many areas of biomedical research, images are crucial for scientific progress. Interactive access to these images is essential for exploration and diagnosis, enhancing understanding, particularly in fields such as pathology, radiology, and cell biology. As imaging techniques continue to advance, generating ever more detailed datasets, the amount of data to be stored and processed will continue to grow. Consequently, data and computationally intensive processes are being increasingly relocated to centralized resources with substantial storage and processing capabilities. However, large, multidimensional and multi-modal biomedical images, such as those generated in experiments with Mass Spectrometry Imaging (MSI), pose a major challenge for fast, comprehensive and interactive remote access. Processes as image data exploration, image analysis, the development of new image analysis methods, and interdisciplinary collaboration of domain experts can be hampered if data-intensive transfers to local systems are required, e.g. for processing of images with interactive applications the domains expert are familiar with. Current efforts to utilize remote resources focus on providing integrated environments for remote development and applications for execution of reproducible image analysis workflows, while lacking comprehensive interactive capabilities with those images.

The methods presented in this work are the results of an inspiring collaboration with members from the research projects "Multi-modal Analytics for the Life Science - Pharmaceutical - Chemical Industry" (M²Aind) and "Mannheim Molecular Intervention Environment" (M²OLIE). A core objective of M²Aind is the development of new image analysis methods and the analysis of molecular features of biological samples. Structural and molecular imaging features are captured using different imaging modalities, including molecular imaging such as MSI and various optical imaging methods such as Fourier-Transform Infrared Spectroscopy (FTIR), Immuno-Fluorescence Microscopy (IF), Immunohistochemistry (IHC) and e.g. light microscopy imaging of Hematoxylin and Eosin (H&E) stained samples. The aim of the project M2OLIE is to explore, conceptualize and establish optimized treatment cycles for cancer patients (oligometastatic liver cancer) in order to accompany them as efficiently and purposefully as possible in the shortest possible time from examination, diagnosis and up to an intervention. Important aspects of this treatment cycle are the initial multi-modal imaging, subsequent minimally invasive, robot-guided biopsies and molecular characterization of the tissue samples. The multi-modal imaging strategy includes Magnetic Resonance Imaging (MRI), Cone-Beam Computed Tomography (CBCT), and Computed Tomography (CT) for initial diagnosis, biopsy planning, and intervention support. New interaction and visualization tools are required for the process of deformable multi-modal image fusion and examined for their integrability in clinical routine. The molecular characterization process of the biopsies includes

molecular imaging techniques such as MSI. The molecular information obtained has the potential to differentiate tumours and tumour subtypes in order to improve diagnostics and individualized patient treatment.

M²Aind and M²OLIE include several sub-projects targeting a broad range of scientific questions based on different imaging modalities that generate a significant volume of data and require interdisciplinary development of new or adapted image analysis methods. The barrier of shifting the image data analysis process entirely to centralized resources could be substantially reduced if a software ecosystem with all required components such as solutions for remote development, remote execution of analysis workflows, remote interactive image exploration, and remote collaboration of domain experts was established. Such a software ecosystem would allow researchers to focus on the scientific question, by enabling seamless use of centralized computing resources, faster interactive exploration of input and output data, and avoiding the burden of repeated data transfer for computationally intensive analysis or collaboration. In addition, specifically regarding MSI, interactive applications of such an ecosystem must be tailored to the requirements of the rapidly developing field of multi-modal and large-scale MSI experiments.

Working with images requires a wide range of processing and visualization capabilities offered by typical image processing applications. These features include the ability to adjust visualization parameters such as intensity ranges and color representations. Additionally, the applications are equipped with navigation tools, enabling detailed examination. The complexity increases when handling hyperspectral and 3D imaging data, due to their additional dimensions. This necessitates more substantial controls, such as selecting 2D projections from hyperspectral or 3D images, to achieve suitable visualizations. Depending on the complexity and size of the images to be processed, comprehensive interactivity is limited in remote scenarios. The familiar interactive desktop applications are no longer available to the user and repetitive data transfers from remote to local systems are often impractical depending on the size of the data. However, experience has shown that a lack of comprehensive interactivity makes it difficult to carry out a more detailed inspection and quality control of imaging dataset as a whole and that a superficial examination of the data can lead to misinterpretations of input or output data, making troubleshooting time-consuming and costly.

Remote working can already be implemented with a variety of tools, including terminal or remote desktop applications that allow users to access and control a computer or virtual machine from another location via a network connection. There are also tools that address remote developing by providing specialized graphical user interfaces and targeted user experiences. Several open-source and web-based Integrated Development Environments (IDE) such as JupyterLab (Jupyter Trademark)[i], RStudio Server[ii], and Visual Studio Code (VS Code)[iii] provide powerful tools for remote development. In contrast, web-based software platforms such as the Galaxy Project[iv,1] and the Joint Imaging Platform (JIP)[2] provide remote access to tools for managing data and to create reproducible processing pipelines. The Open Microscopy Environment Remote Objects (OMERO) offers a specialized data management for biomedical imaging data including a wide range of imaging modalities (e.g. fluorescence and electron microscopy, histological imaging and clinical imaging data). The

[i] https://jupyter.org; accessed April 2024
[ii] http://www.rstudio.com/; accessed April 2024
[iii] https://code.visualstudio.com; accessed April 2024
[iv] https://galaxyproject.org/; accessed April 2024

web-interface of OpenMSI[3] provides fast and convenient access to MSI data, metadata, and derived analysis results stored remotely to facilitate high-performance data analysis and enable implementation of web-based data sharing, visualization, and analysis. Projects such as Dugong[4] provide a monolithic virtual machine-like desktop environment based on Docker[i] specialized for biomedical data analysis. However, setting up and maintaining such platforms can be difficult, and implementing new features can require a high level of expertise due to the complexity of the framework. In context of biomedical imaging in multi-modal setups, the above approaches are

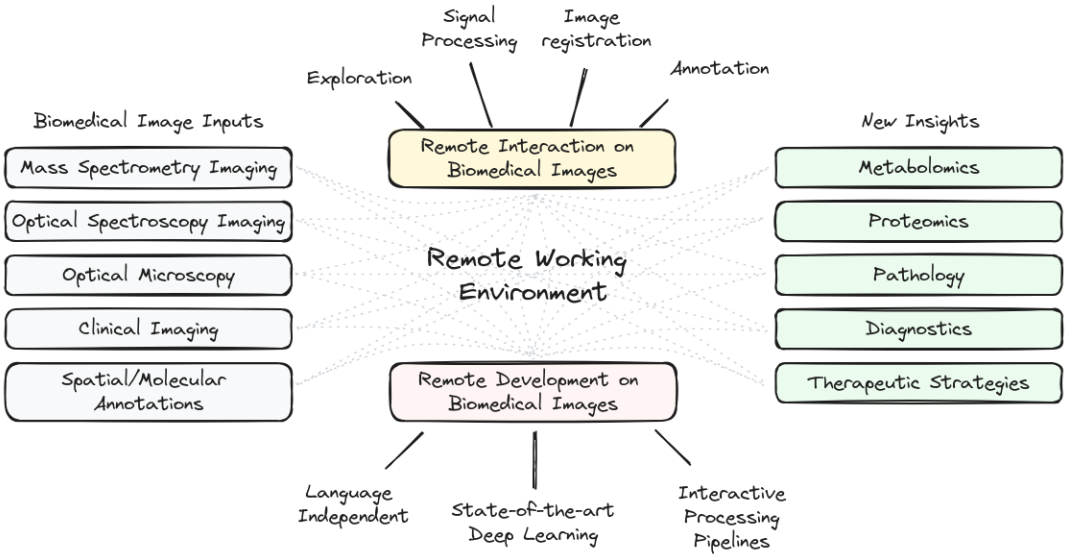

FIGURE 1.1. The interactive remote working environment for the processing and development of biomedical imaging applications.

either limited in their capabilities to instantaneously view, process, or share images interactively at remote sites, and are often rather complex to install, manage and maintain. These limitations indicate a need for a more general, easy-to-maintain concept for remote working that provides access to interactive tools and is adaptable to individual needs and custom tasks. In broad terms, these concepts need to cover a diverse landscape of requirements for the generation of new insights. This is illustrated in Figure 1.1 *Overview of an Interactive Remote Working Environment* and includes (i) support for heterogenous biomedical imaging data and file formats; (ii) remote image analysis and visualization including interactive tasks like exploration, multi-modal image registration, and the creation of spatial annotations; and (iii) remote development capabilities, preferably programming language independent, that integrate with the remote interaction capabilities, facilitating use of remote storage and computing resources for large data and compute intensive tasks like deep learning. In addition, (iv) it can be assumed that a lightweight environment that does not require expert knowledge for installation and maintenance is of advantage. Furthermore, centralized data management processes can greatly facilitate the regular backup, retrieval and sharing of data and strengthen institutional competencies.

Overall, the benefits of integrating remote resources into workflows and establishing them as a standard in research institutes often remain unrealized. Opportunities to save costs, make better use of data resources and establish closer collaboration between different disciplines and individuals such as physicians, experimenters and computer scientists are missed.

[i] https://docs.docker.com/engine/; accessed April 2024

1.2 Objectives

The main goal of this work is to lay the foundations for an advanced software ecosystem that supports interactive processing and remote analysis of biomedical imaging data, especially in the context of MSI. It introduces concepts for the unique challenges of interactive processing of MSI data, which include large data sets, handling multiple modalities, and the need for efficient and scalable solutions for data access and processing. To differentiate from existing solutions, lightweight, flexible and responsive solutions are sought to enable interactive exploration and analysis of multi-modal 2D/3D MSI data. This work places a particular emphasis on providing supporting concepts for the development of deep learning applications in MSI.

1.2.1 Processing multi-modal 2D/3D MSI datasets

A key objective is to create an interactive processing framework that allows for the efficient handling and analysis of multi-modal 2D/3D MSI and related data. The application and development of new methods to analyze MSI often benefit from incorporating imaging data from other modalities. This integrated approach provides additional contextual information that MSI alone might not offer. It enhances the perception of spatial features and allows researchers to analyze molecular conditions in the context of complementary imaging modalities. Furthermore, it facilitates the creation of spatial annotations, which are often required, e.g. in the context of biomarker identification or for the evaluation of newly developed applications and methods, for example in the context of supervised machine learning.

Despite advances in MSI technology, such as increasing spatial resolution, mass resolution and acquisition speed[5], there is a notable lack of open-source applications that provide interactive processing of multi-modal 2D and 3D MSI data[6,7]. This gap underscores the growing need for a user-friendly interactive, open-access software solution that facilitate the integration of advanced algorithms for MSI as well as their development.

Therefore, novel interactive concepts tailored to 2D and 3D MSI datasets in multi-modal setups are presented. This includes a fast raw data import, which is essential for interactivity, fast data processing, and interactive utilities for multi-modal image fusion. Additionally, methods for the interactive evaluation and correction of unsatisfactory fusion results are explored. One challenge is the simultaneous interaction with multiple large 2D MSI datasets for 3D image reconstruction tasks. This requires not only fast, but also memory efficient data import and access structures for MSI datasets. Another challenge is the integration of multiple images from different modalities with diverse image properties, like varying spatial resolutions and pixel types. The goal is to integrate these features into an open-accessible and fully interactive desktop application.

In addition, concepts to support working with deep learning approaches on MSI data are proposed. As deep learning in this area is still at a relatively early stage, dedicated, fast and memory-efficient data interfaces to MSI datasets and related data are missing. The proposed concepts are intended to optimize the data preparation, training and inference processes of deep learning models in MSI, and to be used in combination with the interactive methods outlined above. The hypothesis is that development of DL applications can be improved if interactive processing and visualization of MSI data are applied in combination with scripting for DL. In order to demonstrate the performance of the developed concepts, state-of-the-art deep learning methods from the literature for different MSI analysis tasks, like peak picking and

dimensionality reduction, are adapted to the new interfaces. The developed methods are made openly available to the MSI community so that the implementations of the deep learning approaches can be used as blueprints.

In order to facilitate the combination of the developments based on the aforementioned objectives, another objective emerges: Concepts for seamless integration of script-based DL and other image processing methods into the newly created interactive framework, independent of the programming language the DL script or image processing method is implemented in.

Diverse use-cases and example applications are to be elaborated for the individual objectives to demonstrate the developed concepts.

1.2.2 Interactive Remote Working

It is hypothesized that the interactive, responsive and ad-hoc accessibility of biomedical images on remote resources can significantly accelerate the pace of development and deployment of new scientific methods and innovations. This is especially true for large and processing intensive datasets, such as generated by MSI. With immediate access to remote image data, developers can iteratively test and refine image processing algorithms and tools, receiving immediate feedback on their performance. This rapid research environment is essential for agile research practices, which rely on quick iterations and continuous improvements based on actual user experiences. Additionally, by removing the need to physically handle samples or relocate data to various sites for analysis, remote access minimizes the risk of data corruption and loss, ensuring high integrity and reliability of the data throughout the development process. Furthermore, remote access to images facilitates significantly collaboration among researchers and developers in different locations. By enabling a shared remote access to image data, experts from various disciplines can view, analyze, and discuss the same images in real-time, without the need for physical presence in a lab. This can lead to faster consensus and decision-making, as well as the integration of diverse and interdisciplinary expertise.

This research examines the potential of developing an interactive, fast, and lightweight remote processing and development environment for multi-modal biomedical imaging datasets. The objective is to create an environment that enables work to be carried out as independently of location as possible. This can be achieved by using a central server infrastructure or even by outsourcing to the cloud. The overarching objective is to facilitate interactive access to hyperspectral and biomedical two-dimensional (2D) and three-dimensional (3D) image datasets or collectives, thereby enabling the utilization of central resources, such as hard disk space and central processing unit (CPU) and graphics processing unit (GPU) computing capacity.

1.3 Thesis Structure

This thesis is comprised of seven sections. Following this introductory chapter, which outlines the motivation for the work presented in this thesis, the theoretical background will be outlined in chapter 2. This includes an overview of the basics of data generation, used software and processing methods in MSI. In Chapter 3, the concepts for processing of multi-modal 2D/3D MSI datasets are introduced. These include interactivity, programming language independent integration of image processing methods into an interactive context, and methods to facilitate developing of deep learning models for MSI. In Chapter 4, concepts for interactive remote working are introduced with a focus on remote analysis of MSI and image related

datasets. The results of the introduced concepts are presented in Chapter 5 and discussed in detail in Chapter 6. The thesis is concluded with a summary of the work presented in this thesis in Chapter 7.

Chapter 2

Background

This chapter presents the necessary background for this work. It is structured as follows: In section 2.1 *Mass Spectrometry Imaging*, a brief overview of MSI is given, including basics such as sample preparation, ionization, mass analyzers, batch effects, data structures and file formats. Furthermore, data processing including signal processing, data compression, spatial co-localization and molecular annotations, and the basics of deep learning methods in MSI are introduced.

The fundamentals of image based registration are introduced in section 2.2 *Image-based Registration*. section 2.3 *Bioinformatics* briefly introduces software packages related to the field of biomedical image processing. The basics of the Medical Imaging Interaction Toolkit (MITK) are introduced in section 2.4 *Medical Imaging Interaction Toolkit* as well as the principles of containerization in section 2.5 *Containerization with Docker*.

2.1 Mass Spectrometry Imaging

Image-based analysis of tissue sections is a fundamental tool in biomedical research. The current development is to complement classical histological imaging methods with methods of spatial molecular imaging. By combining morphological and functional features with molecular features, deep insights into the complexity of biological systems and diseases can be gained[8–11]. Here, Mass Spectrometry Imaging (MSI) stands at the forefront of untargeted molecular imaging techniques, providing a powerful means of spatial mappings of the molecular composition of tissue samples.[12–16]. At its core, MSI spatially maps the distribution of molecules within a tissue section, providing researchers with molecular images (ion-images) that can be used to complement the classical histological imaging methods[10]. It provides spectral information of the analyzed pixels with high chemical specificity. The MSI technique utilizes the principles of Mass Spectrometry (MS) to measure the relative abundance of ions (intensity) for a given mass-to-charge ratio (m/z) and spatial position (pixel) (see Figure 2.1), converting complex mixtures of molecules into spatially resolved mass spectra[12,17,18]. This process results in memory-demanding and high-dimensional datasets (hyperspectral images). Its capability of mapping the spatial distribution of molecular species in a variety of biological samples supports applications in various fields, including biology, environmental science, materials science, pharmacokinetics, toxicology, and personalized medicine[5,17,19]. In spatial metabolomics and proteomics, spatial molecular features of the proteome or the metabolic system are acquired to derive biological insights of pathways and structure of those systems[5,20–22].

MSI is commonly categorized as an untargeted molecular imaging method, but depending on the MSI method chosen, a class of molecules is targeted. This enables the detection of specific morphological or functional spatial features related to the targeted molecular class. Examples of MSI devices and corresponding molecular

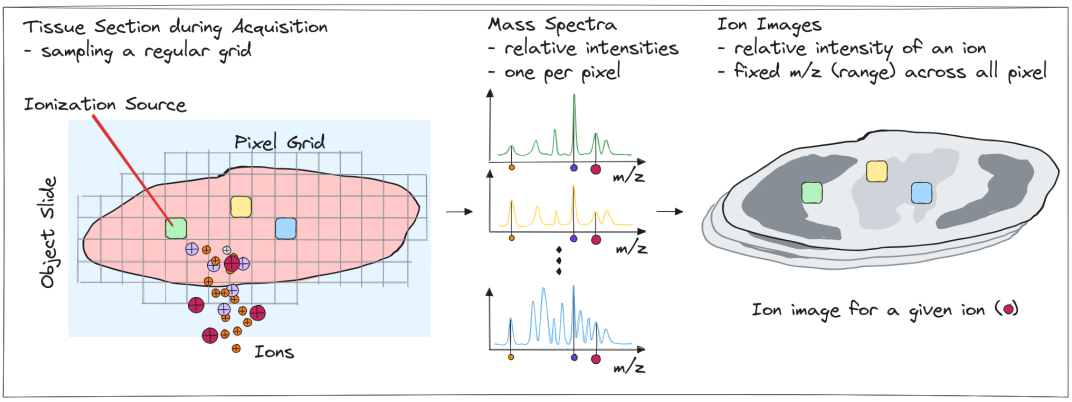

FIGURE 2.1. MSI data acquisition and ion-images. A tissue section is scanned with an ionization source in a regular grid. The relative abundances of molecules are detected for each pixel to generate a plot of mass-to-charge ratio (m/z) values to intensities (mass spectrum). The intensity values of these mass spectra can then be spatially visualized as ion-images, e.g. by mapping intensities or intensity ranges for a specific m/z value (molecule of interest) to the corresponding pixel position.

classes are: Matrix-Assisted Laser Desorption/Ionization (MALDI) Time-of-Flight (TOF) is a versatile mass spectrometry technique capable of analyzing a wide range of biomolecules. It is particularly adept at targeting lipid species, peptides, and proteins within biological samples[8,20]. Desorption Electrospray Ionization (DESI) offers an approach, specializing in the analysis of lipids, metabolites, and drugs[23]. On the other hand, Secondary Ion Mass Spectrometry (SIMS) excels in providing detailed information on elemental distribution and isotope labeling within samples[24] - all methods offering insights into the molecular composition and structure of diverse materials and biological specimens. However, the imaging of a single targeted molecular class may not reflect important morphological features of a tissue section and, therefore, MSI is often used in combination with other imaging modalities for a joined analysis (image fusion). Thus, a variety of imaging modalities is combined with MSI, including optical microscopy or spectroscopy imaging techniques like FTIR, IF or Imaging Mass Cytometry (IMC), as well as clinical modalities such as MRI or CT[25,26]. Furthermore, the collection of multiple MSI datasets focusing on different classes of molecules are also used to obtain a feature-rich representation of tissue samples[27]. The main challenge in image fusion is the transfer of images into a shared image space to spatially align structural and molecular image features. Image fusion includes the application of image-based registration methods, that are substantial research subject in MSI and other disciplines[6,28,29]. A related topic in MSI, is the 3D image reconstruction. Image-based registration methods are used to arrange several neighboring 2D MSI images of the same sample in such a way that a digital volumetric MSI dataset is created. The result of this process can then be used to study molecular features in their three dimensional (3D) environment. Various strategies for image reconstruction, visualization and analysis of 3D MSI datasets have already been applied[7,30–34].

The use of MSI techniques will increase significantly in future applications due to advances in acquisition speed and device precision, resulting in ever-increasing amounts of data, often exceeding several tens of gigabytes for a single MSI dataset. MSI technology not only enables a new understanding of biological processes in their spatial context, but also poses new challenges for the interpretation and management of the complex, high-dimensional and memory-intensive datasets it generates.

Existing open software packages are often limited by time-demanding image initialization procedures, computer memory-intensive implementations, and a lack of interactive capabilities, especially for simultaneous visualization of multiple images and modalities (MSI-to-MSI; MSI-to-Microscopy) in a shared space and for displaying a potential high number of 2D MSI datasets for 3D image reconstructions. In this context, taking a closer look at required capabilities of interactive applications for image-based registration tasks in MSI seems promising for several reasons:

- Interactive applications allow researchers to fine-tune the registration process by visually inspecting the alignment of images[35]. This level of control can significantly improve the accuracy and precision of image registration, ensuring that image features are correctly aligned across different datasets.
- MSI datasets often exhibit variability in terms of sample preparation, tissue morphology, and imaging conditions[36]. Interactive applications provide flexibility to adjust signal processing and registration parameters based on the specific characteristics of each dataset.
- Registration algorithms may struggle with certain types of datasets or regions within the sample. Interactive applications allow researchers to quickly identify and correct misalignments in problematic areas.
- Researchers often possess valuable domain-specific knowledge about the biological samples being studied[37]. Interactive applications empower researchers to leverage their expertise by incorporating qualitative assessments and domain-specific criteria into the registration process, further improving the accuracy and reliability of the results.
- Interactive applications serve as valuable tools for training and education in the field of MSI. By allowing users to interactively explore the registration process and visualize the effects of parameter adjustments, these applications facilitate hands-on learning and skill development for both novice and experienced researchers.

With the future development of instrumentation, data analysis and methodology in the field of MSI, the possibilities for its application in medical and biological research are promising. However, challenges such as data complexity, standardization and integration with other imaging modalities remain. By overcoming these obstacles, MSI could further revolutionize our understanding of the molecular characteristics of disease and biological processes. Based on the lack of open-source software solutions for image-based registration tasks, there is an increasing demand for user-friendly and fully interactive software solutions.

The fundamentals of MSI are introduced by starting with sample preparation and device specificities in subsection 2.1.1 *Fundamentals*, batch effects in subsection 2.1.2 *Batch effects*, the data processing in subsection 2.1.3 *Data Processing* and deep learning methods in subsection 2.1.4 *Deep Learning for Mass Spectrometry Imaging*.

2.1.1 Fundamentals

Sample Preparation

Proper sample preparation is crucial for analytical success, especially in MSI. Even subtle differences in sample integrity or molecular density can significantly impact the measured signal intensity, types of molecules detected, and spatial mapping in

MSI experiments[17,36]. Before analysis, samples are prepared to facilitate ionization and improve detection sensitivity. Figure 2.2 illustrates the sample preparation steps for MALDI MSI. These steps include the collection of tissue samples followed by a procedure to stop enzyme activity to reduce degradation and delocalization of molecules. This is done by flash-freezing or Formalin Fixation Paraffine Embedded (FFPE). Next, thin sections (6-20 µm thickness) are mounted on an appropriate surface (e.g. microscopy slide). Further details on the sample preparation can be found in Buchberger *et al.* (2018)[1717]. In MALDI MSI, for example, a matrix is applied to the sample, promoting the desorption and ionization of molecules upon laser irradiation[12,15,38].

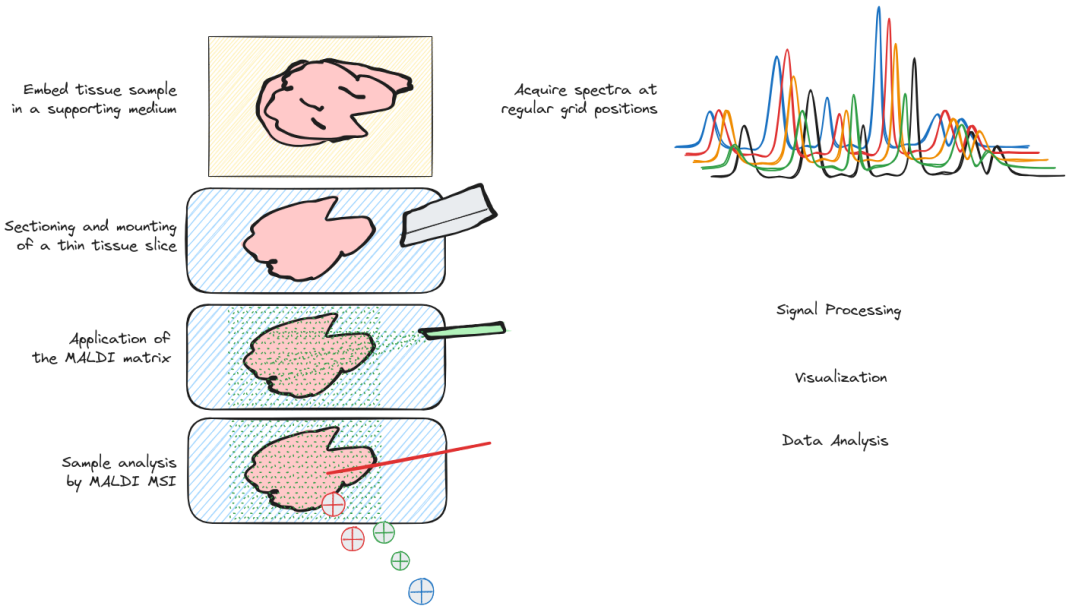

FIGURE 2.2. Illustration of Sample Preparation Steps for MALDI MSI: The process begins with the embedding and mounting of the tissue sample, followed by the application of the matrix. This sequence sets the stage for the MALDI MSI technique (First Column). Subsequent steps involve the acquisition of spectra, signal processing, visualization, and comprehensive analysis of the data (Second Column)

Ionization

The mass spectrometer can be separated in three parts. The *ion source*, the *mass analyzer*, and the *detector*. Commonly used ionization methods for different surfaces are SIMS, MALDI MSI, and DESI. The ionization method determines the spatial resolution and type of analytes that can be detected[38]. In SIMS a focused primary ion beam is used to irradiate, e.g. solid surfaces or thin films, analyzing ejected secondary ions[39]. DESI imaging utilizes the application of an electrically charged solvent mist (electrospray) causing ionization and desorption of surface molecules[40]. On the other side, MALDI MSI, uses a laser beam to irradiate biological tissue surfaces which have being homogeneously coated with a matrix across the tissue surface. The matrix crystallizes together with tissue bio-molecules (co-crystals) and helps to absorb the laser energy to support molecular ionization and desorption from the surface. Depending on the chosen metrices, different classes of bio-molecules can be targeted[41]. All ionization methods commonly generate a cloud of ionized molecules which are then transported into the mass analyzer inlet of the MSI device.

Mass Analyzer

The mass analyzer can measure ionized molecules based on there m/z property. Different types of analyzers exists, including TOF and Fourier-transform ion cyclotron resonance (FTICR)[42]. A TOF mass analyzer measures the time it takes for ions to travel a known distance. By determining the time of flight, the m/z ratio of ions can be accurately calculated, enabling precise mass analysis in a wide range of applications[41]. A detailed view on a MALDI based TOF analyzer is illustrated in Figure 2.3. FTICR mass analyzer utilizes a strong magnetic field to trap ions in a cyclotron motion. By measuring the frequencies of the ion cyclotron resonances, FTICR provides high-resolution mass spectrometry data, enabling accurate mass determination and structural analysis of molecules[43].

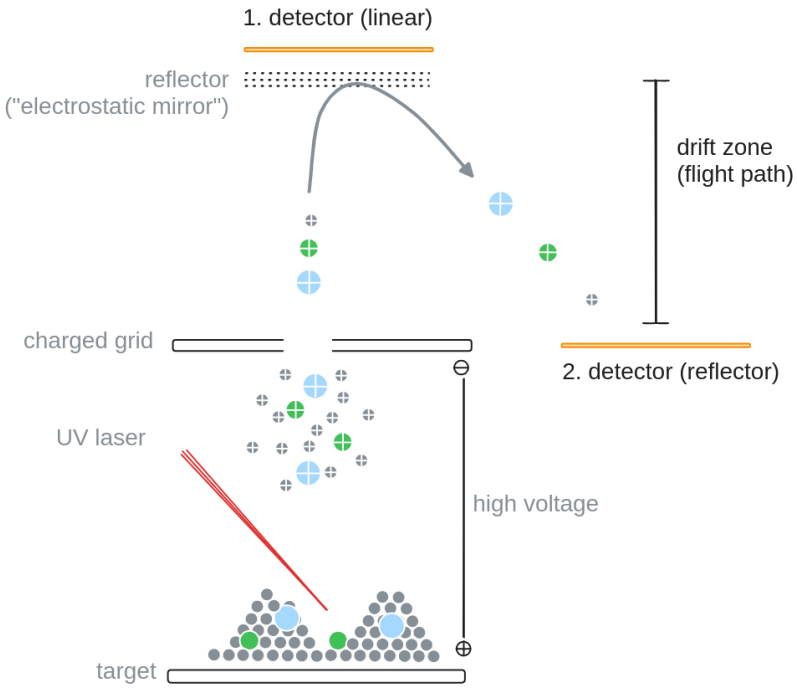

FIGURE 2.3. Drawing of the principal mechanism of a MALDI-TOF device in positive ionization and reflector mode. The basic principle of matrix-assisted laser desorption/ionization time-of-flight (MALDI-TOF) process occurring in the mass spectrometer for positive ionization. The mass resolving power can be improved by further extending the flight path using an electrostatic mirror (reflector). Illustration created by Leopold et al.[41] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0)

Figure 2.3 illustrates the principle mechanisms of a MALDI-TOF device. Positive charged ions are accelerated towards a detector utilizing an electric field. After acceleration those ions move in a field-free space (drift-zone), where the separation of ions is achieved depending on their m/z. Lower mass ions reach the detector faster (linear mode). The peak resolution of TOF mass analyzers can further be enhanced by extending the flight path of the ionized molecules. This can be realized by reflecting ions to a second detector (reflector mode)[41].

Predominantly, MALDI MSI is used in the biomedical research due to its capability of generating high spatial resolution images and its flexibility in the bio-molecular classes to be investigated[13,17]. It is considered as a soft ionization process with minimal fragmentation of the analyzed bio-molecules[44].

Mass Resolving Power

Different mass analyzers can be specified by their Mass Resolving Power (MRP). This is a crucial parameter in analytical techniques like mass spectrometry, denoting the instrument's capability to distinguish closely spaced masses of ions. It is defined as the ratio of the mass of a peak in the mass spectrum to the difference in mass between two peaks that are just resolved. A higher MRP signifies a superior ability to separate ions with similar masses. The MRP is defined as $MRP = m/\Delta m$, where m represents the mass of the peak of interest, and Δm is the minimum mass difference between two peaks that can be resolved[45]. The common MRP for a TOF device lies in the range between 10k-50k[46] and for FTICR devices at m/z 400 greater than 1.5 million[47].

2.1.2 Batch effects

There are many effects that influence the quality and content of the MSI results and are related to the technical implementation of the MALDI-MSI[11]. These effects, which currently represent the greatest challenges in the field of MALDI-MSI, include ion suppression, the homogeneity of the matrix used, enzymatic processes, and the delocalization of analytes. Additionally, validated and standardized protocols are missing or just in the early stage of development[13,36].

On top of these effects, the so called *Batch Effects*, already known from other omics disciplines[48] and digital pathology[49], are influencing the reliability and interpretability of MSI results. Batch effects are effects that accumulate and inflate the variance or introduce noticeable differences between samples. Special batch effects related to MALDI MSI specifically were introduced by Balluff *et al.* (2021)[3636]. Different scales or layers of batch effects were identified, reaching from pixel, section, and slide to time and location related factors.

MALDI mass spectrometry (Matrix-assisted Laser Desorption/Ionization Mass Spectrometry Imaging, MALDI-MSI) is a powerful tool for analysing the spatial distribution of molecules in tissue samples. However, its utility can be compromised by batch effects, which manifest as variations in signal intensity and quality between individual pixels and tissue sections. Pixel-to-pixel batch effects are primarily due to sample preparation. The application of the MALDI matrix results in uneven distribution and crystallization, which leads to spatially uneven suppression of the analytes. In addition, differences in tissue properties and biological content can affect ion extraction and ionization efficiency, which is particularly pronounced with MALDI-based ionization. Acquisition biases, such as decreasing detector sensitivity or matrix evaporation during longer sessions, exacerbate these batch effects. In addition, uneven sample topology and stage tilt contribute to mass shifts between pixels, affecting mass resolution and accuracy. Discrepancies in tissue topology also lead to spatially varying laser focus and ablation spot sizes, resulting in non-uniform ionization.

Section-to-section batch effects have a similar origin to pixel-to-pixel effects, but are specific to differences between tissue sections positioned at different locations on the measurement slide. Variations in laser irradiation and/or matrix application between sections as well as differences in tissue section thickness, processing and storage contribute to systematic differences. In addition, these effects are compounded by laboratory-level variability resulting not only from differences in the preparation of individual pixels and sections, but also from overall sample preparation protocols and time-related factors. These cumulative batch effects can obscure actual biological responses, which is particularly problematic in biomarker discovery applications where ion intensities are highly sensitive to perturbations.

To overcome these challenges, much research has focused on improving the reproducibility of MSI experiments. Studies have compared the results of clinical tissue samples from different facilities and sites using standardized protocols[50]. Hardware advances have also been made, including modifications to the optics to correct for height differences and the development of new hardware configurations that enable MALDI-based imaging of tissue under atmospheric pressure conditions[51,52]. The overall goal of these efforts is to reduce batch effects and technical variability and ultimately improve the reliability and interpretability of MALDI-MSI data for biomedical research and clinical applications.

Data Structure and File Formats

Mass spectrometry imaging (MSI) datasets typically contain two main components: spatial information and mass spectra.

The spatial information component describes the spatial coordinates of each data point within the sample being analyzed. It is represented as x-y-z coordinates of the sampled positions of a regular grid. This information allows to visualize the spectral information in the spatial domain.

For each spatial coordinate, a mass spectrum is recorded. A mass spectrum is a plot of intensity versus mass-to-charge ratio (m/z) and represents the ions detected at that particular location. Each peak in the mass spectrum corresponds to a specific molecule or fragment, and its intensity indicates the abundance of that molecule or fragment at that location.

MSI datasets can be stored in various formats, including proprietary formats specific to the instrument manufacturer, as well as open-source formats such as imzML (imaging mass spectrometry markup language)[53]. These formats typically include metadata describing experimental parameters, such as instrument settings, sample preparation methods, and data processing steps, in addition to the spatial information and mass spectra.

MSI acquisitions usually result in memory-intensive hyperspectral data sets, also known as hyperspectral data cubes. Depending on the MSI device, the number of different values on the m/z axis for the individual spectra (bins) may range from 10^3 for TOF devices to over 10^4 for FTICR[20]. Device manufactures often include a centroiding strategy (the process of reducing a profile spectrum into a spectrum containing only peak information) into their data acquisition pipelines, that reduces the final data size and makes data storage and processing easier, especially for analyzers with very high MRP analyzers such as FTICR. Figure 2.4 illustrates these different spectrum types.

MSI datasets are complex collections of information generated by the spatially resolved detection of molecules in a sample. These datasets typically consist of mass spectra acquired at multiple locations across a sample surface. In two-dimensional MSI datasets, each spectrum corresponds to a spatial point in two dimensions, while in Three-Dimensional (3D) MSI datasets, additional dimensions such as depth are considered. Open file formats like imzML[53] are commonly used for storing MSI data, providing compatibility across different software platforms. Vendor-specific formats, such as file formats for Bruker or Thermo Fisher instruments, are also prevalent. MSI data sets contain not only mass spectra and spatial coordinates but also potentially additional metadata describing the sample and experimental conditions. Collaboration and standardization efforts aim to improve data sharing and reproducibility in the MSI field, facilitating the exchange of information between researchers and advancing our understanding of complex biological systems.

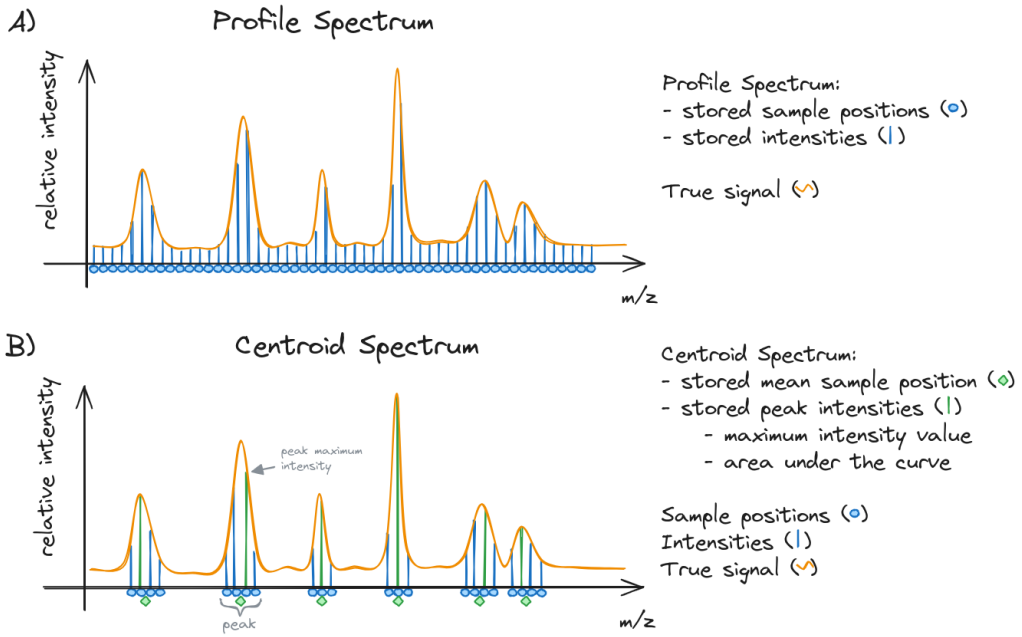

FIGURE 2.4. Conceptual illustration of profile and centroid spectrum types as they were stored in MSI datasets in each pixel position. For profile spectrum (A) the intensity values at each sampling position on the m/z axis are stored. For centroid spectra (B) only the mean m/z value and the peak intensity is stored. Original illustration.

2.1.3 Data Processing

Signal Processing

A fundamental step for accurately reporting the identity and quantity of observed molecules in MSI datasets is the spectrum-wise processing of the acquired mass spectra. Signal processing methods are used to remove noise and artifacts from the spectral information. A typical workflow includes baseline correction, spectral binning, smoothing, mass alignment, normalization, peak picking, and peak binning. Signal processing methods have been discussed in detail by Buchberger *et al.* (2018)[17][17], Ràfols *et al.* (2016)[18][18] and Hu & Laskin (2022)[54][54]. The following listing provides a brief overview of approaches for the different steps of a signal processing pipeline.

- **Baseline Correction** methods are utilized to remove unwanted background signals and distortions from spectral data. These methods involve modeling and subtracting baseline signals, which may arise from factors like electronic noise, chemical interferences, or instrumental artifacts. Common techniques include sliding-window based filtering, polynomial fitting, asymmetric least squares smoothing, and wavelet-based methods, which aim to accurately estimate and remove baseline components.

- **Spectral Binning** involve grouping m/z values into discrete bins to simplify and reduce the dimensionality of spectral data. By aggregating adjacent m/z values into bins, spectral binning improves signal-to-noise ratio and facilitates downstream analysis such as peak picking and statistical comparisons.

- **Spectral Smoothing** methods are employed to reduce noise and enhance the quality of spectral data. These methods typically involve applying mathematical filters or algorithms to attenuate high-frequency noise while preserving underlying molecular signals (peaks). Common techniques include moving average, Savitzky-Golay[55], Gaussian smoothing, and approaches including neighboring spectra. Spectral smoothing helps to enhance the visualization and interpretation of molecular features, particularly in complex MSI datasets. However, careful consideration must be given to the choice of smoothing parameters to avoid distortion or loss of important spectral information.

- **Mass Alignment** aims to correct for mass shifts in the m/z values across spectra, ensuring accurate spatial alignment and comparison of molecular features. These methods typically involve aligning peaks or mass channels across spectra by applying mathematical transformations such as interpolation or warping. Peak-based alignment techniques identify common peaks in spectra and adjust for mass shifts to align them. Alternatively, warping-based methods utilize algorithms to deform spectra to match a reference spectrum. Accurate mass alignment is essential for precise spatial mapping and comparative analysis in MSI studies, facilitating the identification of biomarkers and molecular patterns.

- **Spectral Normalization** aims to correct variations in ion intensity across spectra, enabling more accurate quantification and comparison of molecular signals and ion-images. These methods typically involve dividing each spectrum by a normalization factor, often derived from internal standards or reference peaks. One common approach is total ion count (TIC) or root mean squared (RMS) normalization, where each spectrum is divided by its total ion intensity. Another

method involves normalization to a reference peak or set of peaks, ensuring consistency across samples. Spectral normalization helps mitigate technical variability, improving the reliability and reproducibility of MSI data analysis.

- **Peak Picking** methods are employed to identify and extract peaks representing molecular ions from spectral data. These methods involve algorithms that automatically detect peaks based on characteristics such as intensity, shape, and signal-to-noise ratio. Common peak picking algorithms include thresholding, local maximum identification, and wavelet-based methods, which can be tailored to different types of spectra and analytical objectives. Peak picking is crucial for quantifying molecular abundance and identifying biomarkers or molecular features in MSI datasets. However, selecting an appropriate peak picking method requires careful consideration of factors such as spectral complexity, noise level, and desired sensitivity.

Data compression

A MSI dataset is represented by a high-dimensional 3D data cube. Data compression methods include dimensionality reduction techniques, that aim to reduce the complexity of data by transforming high-dimensional mass spectra into lower-dimensional representations[17,54]. One common method is principal component analysis (PCA), which identifies orthogonal components capturing the most variance in the data. PCA can help visualize sample variations and highlight significant features. Another approach is non-negative matrix factorization (NMF), which decomposes mass spectra into non-negative components, aiding in feature extraction and interpretation. T-distributed stochastic neighbor embedding (t-SNE) is a popular non-linear dimensionality reductio technique for visualizing high-dimensional data in two or three dimensions. It preserves local structure and is particularly useful for clustering and identifying spatial relationships in MSI datasets.

Data compression methods also include the spatial grouping of spectra into regions with similar characteristics. Clustering methods group similar mass spectra into clusters based on various similarity metrics[56]. K-means clustering partitions spectra into k clusters by minimizing the within-cluster sum of squares. Hierarchical clustering organizes spectra into a tree-like structure, revealing hierarchical relationships among samples. Density-based spatial clustering of applications with noise (DBSCAN) identifies clusters based on regions of high density, making it robust to outliers. Gaussian mixture models (GMM) assume that the data is generated from a mixture of several Gaussian distributions, allowing for flexible cluster shapes. Self-organizing maps (SOM) are neural network-based methods that map high-dimensional data onto a low-dimensional grid, preserving topological relationships between spectra. Overall, dimensionality reduction and clustering methods play crucial roles in simplifying and extracting meaningful information from complex MSI datasets.

Spatial Co-localization

Spatial co-localization in MSI refers to the detection of two or more ion-images within similar spatial analyte distributions. This technique is particularly useful for studying protein-protein interactions, protein-lipid interactions, and molecular assemblies. A study, published by Ovchinnikova *et al.* (2020)[57] compares different spatial co-localization measures including, Pearson correlation, Spearman correlation and cosine similarity for flattened ion-image pairs (1D vectors of pixel intensities)[57,58].

Molecular annotations

Understanding the molecular composition is essential for extracting meaningful biological insights from MSI data. Traditionally, peak annotation in MSI experiments relies on accurate mass determination obtained either directly from tissue or through off-line bulk analysis. The integration of ion mobility spectrometry with MSI has enhanced isomeric and isobaric separation, providing additional structural information for analyte identification. Furthermore, MS/MS imaging allows for the simultaneous visualization and identification of biomolecules within tissues[17]. Beside improved analysis techniques, community resources have been developed to facilitate the identification of various molecular classes, including metabolites, lipids, glycans, and proteins, utilizing accurate mass, MS/MS, or collision cross-section (CCS) measurements. Database searching and scoring algorithms play a pivotal role in this process. Palmer et al. devised an false discovery rate (FDR)-controlled metabolite annotation method specifically tailored for MSI data, incorporating accurate mass, isotopic pattern, and spatial distribution into a joint match score to enhance search performance[59]. Additionally, rMSIannotation utilizes isotopic pattern coherence, image correlation, and mass error for the annotation of isotopic and adduct ions in MSI data[60].

2.1.4 Deep Learning for Mass Spectrometry Imaging

Recent advances in deep learning have revolutionized various fields, particularly in medical image processing. Artificial neural networks (ANNs) and convolutional neural networks (CNNs) have significantly contributed to tasks such as image detection, recognition, segmentation, and computer-aided diagnostics in medical imaging. Deep learning, a subset of machine learning, encompasses supervised, semi-supervised, and unsupervised learning paradigms. Supervised learning involves providing both input and desired output data during training, allowing the algorithm to learn the relationship between them, such as pixel-class pairs. Unsupervised learning involves finding structure in input data without explicit output guidance. Semi-supervised learning lies between these two approaches, incorporating limited prior information. These learning concepts are integral to deep learning, which relies on ANNs inspired by the brain's information processing. By mimicking biological learning processes, deep learning algorithms can extract meaningful structures from input data. Understanding how these algorithms uncover internal representations within data is crucial for their effective application. Deep learning techniques have been increasingly applied in medical and biomedcial image processing due to their ability to handle large datasets and extract complex features. They have shown promise in tasks such as tumor detection, organ segmentation, disease classification, treatment planning, and tissue classification.

The enormous feature density of the spectral information provided by MSI offers an interesting basis for deep learning applications[54]. Compared to other disciplines where deep learning was successfully applied, only a comparatively slow progress for MSI datasets is recognizable. This may be attributed to the lack of benchmark datasets[20], high data variance within and between acquisitions as a consequence of batch effects[36], and a diverse landscape of MSI processing utilities. Additionally, most deep learning code is written in Python, whereas most MSI data science packages are written in R. Feeding data to neural networks is a bottleneck in MSI processing and optimized Python packages for MSI data access and processing are rare. While deep learning offers significant potential for MSI, as in other disciplines, there are still challenges, such as the need for annotated datasets, interpretability of model

predictions, and generalization to diverse subject populations. Addressing these challenges presents opportunities for further research and development in the field[20].

Deep Learning models using spectral data

CNN architectures, particularly utilizing 1D convolution computations, have shown promise in extracting and processing molecular information from correlated mass spectral data. For instance, Behrmann et al. employed convolution kernels of size 3 with a small receptive field to discern isotopic patterns in IsotopeNet[61]. Similarly, dilated convolutions were utilized to capture both locally and globally distributed spectral patterns. Despite their potential, the limited amount of MSI training datasets poses a significant bottleneck to the utilization of deep learning approaches. Various strategies have emerged to address this challenge. One such strategy involves the use of multiple manually annotated datasets to train the same representation learning model in a supervised manner. Seddiki et al. implemented a cumulative learning strategy, sequentially training the same CNN model with MSI datasets acquired from different organisms, optimizing the model parameters for learning spectral representations from diverse MSI data[62].Building upon the success of msiPL[63], Abdelmoula et al. developed massNet[64], utilizing the msiPL autoencoder to train the neural network for learning mass spectra representations. Leveraging the pretrained encoder, they trained a classification model to distinguish between tumor and healthy tissues.

Deep Learning models using spatial data

Recently, CNN models have become increasingly popular, especially in the study of molecular co-localization. For example, Zhang et al. applied a transfer learning technique to extract spatial features from ion-images using a pre-trained Xception CNN model. Their approach outperformed UMAP in clustering analysis and provided better classification of ion-images[65]. Ovchinnikova et al. tackled this challenge through supervised learning by creating a sizable number of gold standard ion-images manually curated by experts from the METASPACE knowledge base. Using this baseline data, they developed a deep learning-based Pi model to quantify colocalization between ion-images. Furthermore, they trained a ResNet-50 CNN model with a labeled dataset and achieved a 98% recognition rate in identifying ion-images that do not correspond to the sample[57]. Despite the successes of these studies, a notable challenge arises from the limited size of MSI data for traditional deep neural network training frameworks, which necessitate extensive annotated ion-images. To address this challenge, self-supervised learning emerges as a promising approach. Hu et al. introduced a self-supervised clustering method for MSI spatial data compression using contrastive learning and image augmentation[58]. Employing a simple framework for contrastive learning of visual representations (SimCLR[66]), they trained a deep CNN encoder using MSI data from a single experiment without manual annotations. This model effectively learned spatial features from ion-images and classified them based on molecular colocalizations, visualized in a 2D space using t-SNE.

2.2 Image-based Registration

The fundamentals of image based registration methods are introduced in subsection 2.2.1 *Basic Concepts*, followed by an overview of applied algorithms in subsection 2.2.2 *Registration Algorithms* and validation and evaluation strategies in subsection 2.2.3 *Validation and Evaluation*.

2.2.1 Basic Concepts

Image-based registration is a fundamental process in biomedical imaging that involves aligning and merging multiple images obtained from different modalities, time points, or spatial domains. It plays a crucial role in clinical applications such as image-guided surgery, treatment planning, and disease monitoring. Also in biomedical imaging applications, and especially in MSI, these methods are required to create image fusions for co-localized representation of image features and 3D image reconstructions. An extensive summary of image-based registration is given in[29,67,68].

Image registration refers to the spatial alignment of two or more images to establish a correspondence between their pixel or voxel locations. In the context of biomedical imaging, image-based registration involves the fusion of anatomical or functional information from various imaging modalities, including clinical modalities like Computed Tomography (CT) and Magnetic Resonance Imaging (MRI), as well as microscopy modalities like light-sheed, ImmunoFluorescence Microscopy (IF), optical tissue scanners, con-focal light microscopy, or spectrometry/spectroscopy modalities like Mass Spectrometry Imaging (MSI), Fourier-Transform Infrared Spectroscopy (FTIR), Raman spectroscopy[6].

Image registration involves finding an optimal transformation T that aligns the images (Transformation Models). This is illustrated in Figure 2.5 *Optimization process in image based registration*. Commonly used transformation models include rigid, affine, deformable, and non-rigid models. Rigid transformations preserve shape and do not allow for local deformations, while affine transformations allow scaling, rotation, translation, and shearing. Deformable models provide more flexibility by allowing local deformation. Non-rigid models offer even greater freedom by allowing non-linear deformations.

To quantify the alignment between images, similarity measures are used. These measures assess the correspondence between pixel or voxel intensities, image gradients, or higher-order image features. Commonly used similarity measures include Sum of Squared Differences (SSD), Normalized Cross-Correlation (NCC), and Mutual Information (MI). The choice of similarity measure depends on the imaging modality, image characteristics, and registration goals[35,69,70]. Multi metric approaches are used to optimize the registration problem with two or more similarity measures. Most of the similarity measures are based on corresponding characteristics of the fixed and moving image. To help the registration, corresponding point pairs in the fixed and moving images can be defined. A metric that minimizes the distance of two point sets with known correspondence can be used to help in a difficult image registration task that fails if performed fully automatically.

2.2.2 Registration Algorithms

Intensity-based registration methods use the similarity measures to optimize the transformation parameters iteratively. These approaches aim to minimize the dissimilarity between images by adjusting the transformation parameters until convergence is reached. Optimization techniques such as the gradient descent method are commonly employed. The accuracy and efficiency of these algorithms depend on the chosen optimization strategy, initialization, and convergence criteria.

Deformable registration methods are used when significant morphological (i.e. anatomical or pathological) variations exist between images. These algorithms aim to model and incorporate spatial deformations into the registration process. Common approaches include B-spline-based registration, free-form deformation, or deformable image registration based on biomechanical models. These methods often employ

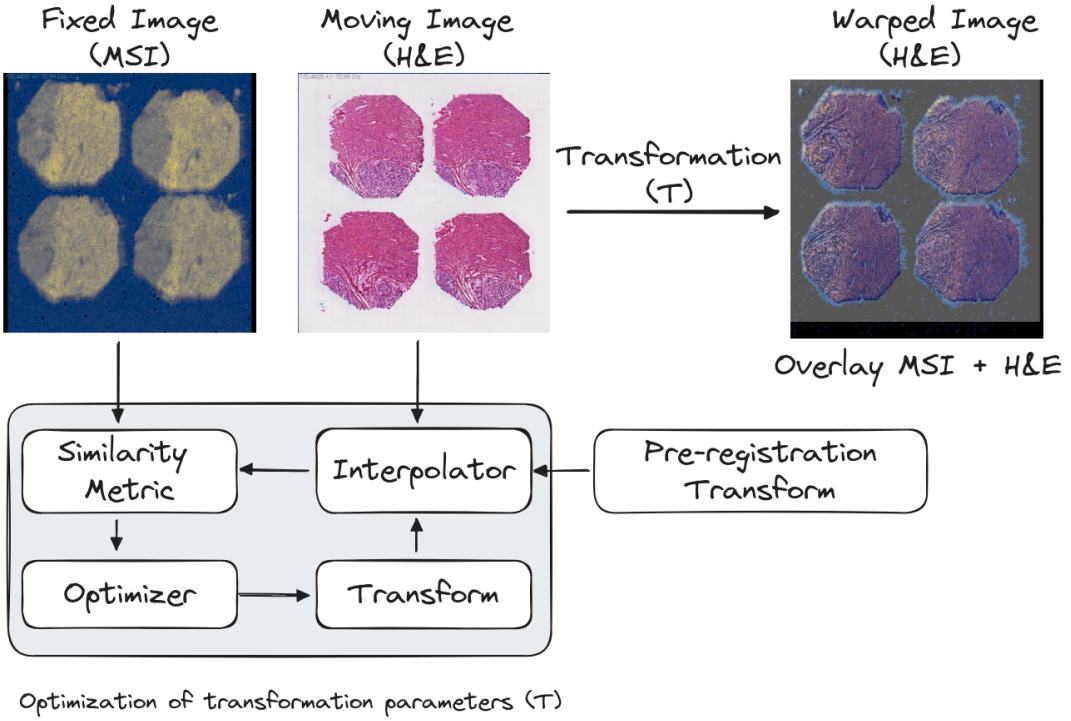

FIGURE 2.5. Schematic illustration of the optimization process of a transformation T so that corresponding features in the images overlap. The depicted H&E stained images and DESI-MSI dataset was published by Oetjen *et al.* (2015)[33] and is available in the MetaboLights repository [MTBLS282][ii].

regularization techniques to balance the smoothness of deformation and the goodness of fit to the data.

Many deep learning-based methods for image-based registration have been developed in recent years. With methods based on deep learning, the iterative optimization of the transformation parameters during the application of these models is no longer necessary; instead, the optimization of the model parameters for a transformation is shifted to the training phase of these models. This leads to considerable time savings in the inference phase. A detailed summary of deep learning models for image-based registration can be found in Boveiri *et al.* (2020)[7171].

2.2.3 Validation and Evaluation

Validation and evaluation of registration results are critical to ensure the accuracy and reliability of the registration process. Various validation metrics, such as target registration error (TRE), landmark-based evaluation, or overlap measures, can be used to quantify the registration accuracy. Additionally, visual assessment and expert judgment play a crucial role in assessing the quality of registration results[35].

2.3 Bioinformatics

A variety of software tools have been developed to help researchers and healthcare professionals in analyzing, interpreting, and visualizing complex biomedical imaging datasets. These tools cater to specific modalities such as MSI, microscopy imaging, and clinical imaging data. Each software tool is used for specific purposes within its respective domain, facilitating analysis, interpretation, and visualization of

biomedical imaging data. Overall, biomedical imaging analysis plays a crucial role in advancing research and enhancing diagnostic and therapeutic capabilities in the field of biomedicine.

A brief overview of what biomedical images are, is given in subsection 2.3.1 *Biomedical Images*, followed by a brief introduction of mass spectroscopy imaging applications in subsection 2.3.2 *Mass Spectroscopy Imaging Applications*, packages for MSI data processing in subsection 2.3.3 *Mass Spectroscopy Imaging Packages*, microscopy imaging software tools in subsection 2.3.4 *Microscopy Imaging Applications*, as well as clinical imaging software tools in subsection 2.3.5 *Clinical Imaging*. Additionally, bioinformatics packages for image based registration are introduced in subsection 2.3.6 *Frameworks for Image-Based Registration*. Finally, the Medical Imaging Interaction Toolkit (MITK) is introduced in section 2.4 *Medical Imaging Interaction Toolkit* and containerization in section 2.5 *Containerization with Docker*.

2.3.1 Biomedical Images

In biomedical science, digital images are typically defined as representations of biological specimens or medical data captured in a digital format. These images are composed of discrete pixels arranged in an isotropic or anisotropic potentially multi-dimensional grid, where each grid position contains numerical values representing the intensities or color of the corresponding portion of the specimen. These grid positions are called pixels in Two-Dimensional (2D) and voxels in 3D image datasets. The values associated with a grid position can also be a sequence of intensity values (e.g. three values are stored for colored images representing the red, gree, and blue color channels). Hyperspectral imaging methods provide spectral intensity values (e.g. often sequences with more then 10^3 values) for each grid position. In order to visualize and interpret image data correctly, real world properties are stored alongside the pixel intensity values. Pixel/voxel spacing, origin, and dimension are key elements in this context[iii].

- Pixel/voxel spacing refers to the physical distance between the centers of adjacent pixels or voxels in each dimension. This metric is crucial for accurate measurements and rendering, especially in medical imaging where precision matters for diagnosis and treatment planning.
- Origin, in image data, specifies the coordinate in the image that corresponds to a known physical location, typically the corner of the scanning field. This allows the imaging system to map pixel data to a real-world coordinate system, facilitating integration with other spatial data sets.
- Dimension indicates the number of grid positions along each spatial axis and determines the overall size of the image or volume.

Together, these properties ensure that digital image data can be accurately related back to physical space, enabling effective analysis, manipulation, and interpretation in a variety of applications. The describing properties of digital images are illustrated in Figure 2.6 *Schematic Illustration of 2D Images*.

Image Segmentation Segmentations, or spatial annotations, in the context of image processing and analysis refer to the process of dividing an image into multiple segments or regions, each representing a meaningful part of the image. This technique

[iii] https://docs.mitk.org/nightly/GeometryOverviewPage.html; accessed April 2024

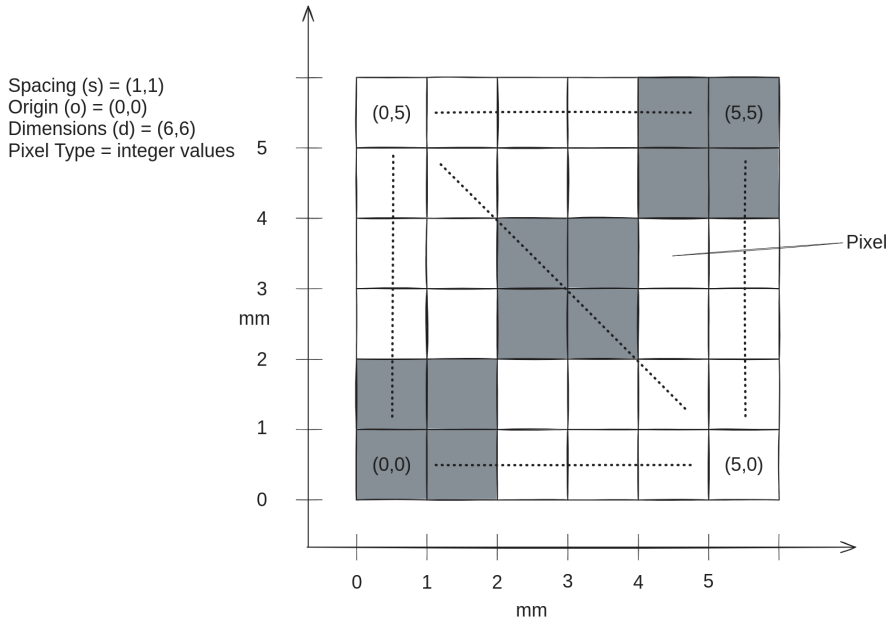

FIGURE 2.6. Schematic illustration of 2D images. The image properties are showing the image dimensions (d), spacing (s) and the same origin (o) in the world coordinate system. In addition the index coordinates of image grid are included in the pixels. Each pixel position provides intensity information in a specific format (pixel type).

is often used to isolate specific objects or features within an image for further analysis, processing, or understanding. Typically, segmentation images use non-floating point intensity values, where each value correlates with specific spatial image features.

2.3.2 Mass Spectroscopy Imaging Applications

Mass spectrometry is a high-throughput measurement methods. These methods perform a large number of measurements in a short time, resulting in a huge amount of data. Bioinformatic software solutions help with the analysis. As the algorithms and data formats used are often not open, the integration of new methods or the evaluation of different approaches is impossible. To address these problems, efforts were made to develop standardized formats and open algorithms that facilitate interoperability and transparency in the analysis of mass spectrometry data.

The steadily increasing influence of MSI in various fields of science has encouraged the development of analysis and processing tools that focus on the use of those hyperspectral and memory-intensive MSI datasets[17,54]. Numerous open-source and vendor-specific software tools have been developed to overcome challenges posed by increasing data volume and improving data quality. Open-source MSI tools and packages in different programming languages, including R, Python, C++, and Matlab, are available targeting the raw data access, signal processing, spatial annotations (e.g. clustering), molecular annotations, dimensionality reduction, image-based registration, and quality control of MSI dataset[59,72–76]. In addition, there are several interactive applications with Graphical User-Interface (GUI) that enable the viewing and editing of MSI data sets[3,18,60,76–86]. Except for the software developed on the basis of the concepts proposed in this thesis, there are still no interactive applications for (semi-)automatic 3D reconstruction or multimodal MSI experiments in an openly accessible and mature form. The currently most advanced commercial solution is SCiLS Lab (Bruker Daltonik GmbH, Bremen, Germany). Open-source toolkits and

command-line applications targeting intensity-based image registration are available, e.g., the Insight Toolkit (ITK)[87] and elastix[88].

rMSI: rMSI is an R package tailored for MSI data analysis, featuring an efficient data management system and an integrated GUI within the R environment[60,78,79]. It utilizes a custom format to minimize memory usage while ensuring fast spectra access. The GUI facilitates easy data exploration and visualization. Additionally, rMSI provides a library of functions for seamless integration with other R packages, enabling convenient sharing of MS data.

OpenMSI: OpenMSI is an open-source platform for visualization, analysis, and sharing of mass spectrometry imaging data[3]. It offers a web-based interface for managing, storing, visualizing, and analyzing MSI data and supports various data formats.

METASPACE: METASPACE is a platform designed for the analysis of MSI data[84]. METASPACE simplifies MSI data interpretation with its user-friendly web interface and advanced algorithms. It enables easy uploading, processing, and analysis, aiding in molecule identification and mapping. The platform offers visualization and analysis tools for deeper insights into molecular composition and supports collaboration through data sharing. Valuable for researchers in biology, medicine, and environmental science, METASPACE facilitates exploration of spatially resolved molecular information.

2.3.3 Mass Spectroscopy Imaging Packages

There are only a few open-source software packages available for the analysis of MSI data. These packages provide tools for data processing, visualization, and analysis, enabling researchers to extract valuable insights from complex MSI datasets. Some of the popular open-source MSI packages include:

Cardinal: Cardinal is an R package tailored for analyzing MSI datasets[73]. It offers efficient tools for signal processing, spatial segmentation, and classification of MSI data.

MALDIquant: MALDIquant is an R package designed for the analysis of MSI[72]. MALDIquant provides non-interactive functions for signal processing, visualization, and analysis of MSI data.

pyImzML: pyImzML[iv] is a Python library specifically designed for handling the import and export of raw imzML data[53]. This package facilitates seamless integration with Python-based data analysis workflows, enabling researchers to process and analyze MSI data.

2.3.4 Microscopy Imaging Applications

A wide range of software tools is available for the analysis of microscopy imaging data, catering to various imaging modalities and research applications. These tools provide functionalities for image processing, segmentation, quantification, visualization, and

[iv] https://github.com/alexandrovteam/pyimzML; accessed April 2024

data analysis, enabling researchers to extract valuable information from microscopy images. Some of the popular (open source) microscopy imaging software tools include:

ImageJ: it provides a user-friendly interface for various image analysis tasks, including image processing, segmentation, quantification, and visualization, particularly in the field of biomedical research and microscopy[89].

CellProfiler: designed for high-throughput image analysis, CellProfiler features a graphical interface for creating and executing pipelines to analyze large sets of biological images, commonly used in cell biology and high-content screening[90].

Icy (Image Analysis Software): Icy provides a user-friendly interface for image analysis and visualization, with a wide range of plugins and tools for tasks such as object tracking, segmentation, and 3D visualization[91].

2.3.5 Clinical Imaging

Clinical imaging software tools are essential for the analysis and interpretation of medical images, aiding healthcare professionals in diagnosis, treatment planning, and patient care. These tools provide functionalities for viewing, processing, and analyzing medical images from various modalities such as MRI, CT, and ultrasound. Some of the popular clinical imaging software tools include:

MITK (Medical Imaging Interaction Toolkit): MITK is an open-source toolkit for the development of interactive medical image processing software[92,93]. It provides functionalities for loading, processing, and analyzing medical images in various modalities such as MRI, CT, and ultrasound. MITK is often used for research and the development of medical image analysis applications.

3D Slicer: 3D Slicer is a free, open-source software platform for medical image informatics, image processing, and three-dimensional visualization[94]. It is widely used in the medical community for tasks such as image segmentation, registration, visualization, and quantitative analysis. 3D Slicer is highly extensible and supports a wide range of medical imaging modalities.

OsiriX: OsiriX is a DICOM viewer for medical images with advanced visualization and analysis capabilities[95]. It is commonly used by healthcare professionals for viewing and analyzing medical images such as CT scans and MRI.

2.3.6 Frameworks for Image-Based Registration

Image based registration methods are available in a wide range of different software packages, including open-source and commercial tools. These tools provide functionalities for aligning and merging multiple images obtained from different modalities, time points, or spatial domains. Image-based registration plays a crucial role in various biomedical imaging applications, including image-guided surgery, treatment planning, and disease monitoring. Some of the popular software tools for image-based registration include:

Elastix: Elastix is an open-source software package for image registration, widely used for its flexibility, efficiency, and ease of integration into existing workflows[88]. Elastix provides various transformation models, similarity measures, and optimization strategies for accurate image alignment. It is commonly used in medical imaging applications for aligning images from different modalities or time points.

ITK (Insight Segmentation and Registration Toolkit): ITK is an open-source software toolkit for image analysis, including image registration, segmentation, and visualization[87]. ITK provides a wide range of algorithms and tools for processing and analyzing medical images, making it a popular choice for researchers and developers in the medical imaging field.

3D Slicer: 3D Slicer provides functionalities for image registration, segmentation, and visualization, making it a versatile platform for medical image analysis[94]. It offers a wide range of tools and algorithms for aligning and merging images from different modalities or time points, enabling researchers and healthcare professionals to analyze complex medical imaging datasets.

The accurate alignment of images is crucial for various applications such as image-guided interventions, disease diagnosis, and treatment planning. Image registration, the process of aligning two or more images into a common coordinate system, plays a fundamental role in achieving this alignment. Among the tools available for image registration, Elastix stands out as a powerful and versatile toolkit widely used for its flexibility, efficiency, and ease of integration into existing workflows.

Transformation Models: Elastix employs various transformation models to align images accurately. These models include rigid, affine, and non-linear transformations.

Rigid transformation: preserves the shape of objects in an image, allowing only translation and rotation. Affine transformation: Adds scaling and skewing to the transformations allowed by rigid transformations. Non-linear transformation: Enables more complex deformations to align images, crucial for capturing non-linear anatomical changes.

Similarity Measures: Elastix utilizes different similarity measures to quantify the alignment quality between images. Common similarity measures include mutual information, normalized cross-correlation, and mean square error. These measures help Elastix optimize the transformation parameters to maximize alignment accuracy.

Optimization Strategies: Elastix implements optimization strategies such as stochastic gradient descent and quasi-Newton methods to iteratively refine transformation parameters. These strategies aim to minimize the discrepancy between images based on the selected similarity measure.

Configuration: Elastix offers a flexible configuration framework, allowing users to define their registration pipeline by selecting appropriate transformation models, similarity measures, and optimization strategies. Configuration files in Elastix are written in a human-readable format, making customization straightforward.

Registration: the registration process in Elastix involves iteratively optimizing transformation parameters to align the moving image with the fixed image. This optimization process is guided by the selected similarity measure and optimization strategy specified in the configuration.

2.3.7 Remote Image Processing and Developing Applications

In this section, applications for remote working with biomedical image data are briefly introduced to provide the reader with an overview of available open source solutions for dedicated remote applicable tasks.

Terminal Applications: applications like SSH (Secure Shell) allow users to access remote servers directly from the command-line. This is often used for processing of large datasets stored on remote computing resources. Users can execute scripts, run software, and manage data without needing a graphical interface, which is especially useful for automated or batch processing tasks.

Remote Desktop Applications: remote desktop applications such as Virtual Network Computing (VNC) and Remote Desktop Protocol (RDP) provide a graphical interface to a remote desktop, offering the full utility of remote systems. These tools are essential for tasks that require interaction with GUI-based applications for image processing, facilitating a seamless workflow as if one were physically present at the remote computer. Main limitations typically refer to the latency issues, lower responsiveness, and sometimes a less intuitive control over applications than when they are run locally. Such issues can interrupt the workflow, especially in tasks requiring high precision, such as detailed image manipulation. The effectiveness of remote desktop solutions heavily depends on the quality and stability of the internet connection. Poor connectivity can lead to disconnections, lag, or even loss of unsaved work, which can be detrimental in critical processes. Setting up a secure and efficient remote desktop environment often requires significant administrative expertise and continuous management, which can be a barrier for smaller teams or institutions without sufficient administrative resources.

JupyterLab: JupyterLab is an interactive development environment that facilitates the creation and sharing of documents that include live code, equations, visualizations, and narrative text. Especially useful for collaborative research, it supports a multitude of data science and imaging libraries in Python such as NumPy[v] and Matplotlib[vi], allowing for on-the-fly analysis and visualization of biomedical imaging data[96].

RStudio Server: RStudio Server enables users to run R and RStudio directly within a web browser, providing a powerful platform for statistical analysis and graphics. This server-based version is particularly beneficial for remote collaboration and accessing R's powerful suite of tools for image processing and data analysis, ensuring that computational resources are centralized and more easily manageable[97].

[v] https://numpy.org/; accessed April 2024
[vi] https://matplotlib.org/; accessed April 2024

Visual Studio Code: Visual Studio Code[vii], (VS Code) with its remote development extension, allows developers to use VS Code on their local machine to connect to and edit code stored on remote servers, containers, or Windows Subsystem for Linux (WSL). This setup is particularly advantageous for developing and debugging image processing applications that need to run close to the data residing on remote servers, minimizing data transfer times and enhancing security.

Joint Imaging Platform (JIP): the Joint Imaging Platform (JIP) offers a suite of tools designed specifically for collaborative medical image analysis and processing[2]. JIP enables seamless integration of image data with analytical tools and supports remote access, allowing researchers to work on complex imaging datasets from anywhere, fostering collaboration across different institutions.

Open Microscopy Environment Remote Objects (OMERO): OMERO is a client-server software platform that enables users to visualize, manage, and share biological image data remotely[98]. Particularly relevant in multi-user, multi-site projects, OMERO handles a wide range of bio-imaging data formats, providing an indispensable tool for remote collaborations in biomedical research.

Galaxy Project: the Galaxy Project provides a web-based platform for accessible, reproducible, and transparent computational biomedical research[1,76]. Galaxy facilitates data integration and provides tools for image analysis, which are accessible via a web browser without the need for programming knowledge, making it ideal for educational purposes and democratizing access to sophisticated data analysis tools.

Dugong: Dugong is an open-source tool designed to provide a reproducible and Docker-based environment for scientific software, including image processing[4]. It offers a containerized solution that ensures consistency across different computing environments, making it highly suitable for developing, testing, and deploying biomedical imaging applications and workflows remotely.

2.4 Medical Imaging Interaction Toolkit

This section introduces Medical Imaging Interaction Toolkit (MITK), an open source software and modular development framework for interactive medical imaging applications. MITK, developed by the German Cancer Research Center (DKFZ), is a versatile platform designed for the processing, visualization, and analysis of medical image data. This includes 2D/3D imaging techniques like MRI, CT and ultrasound[92,93,99]. It provides a robust infrastructure for creating interactive medical imaging applications that meet the complex requirements of healthcare professionals and researchers.

2.4.1 2D/3D Visualization and Rendering

MITK offers advanced 2D/3D visualization and rendering tools for interactive exploration and analysis of medical images. The Multiplanar Reconstruction (MPR) capabilities enhance visualization, allowing users to view cross-sectional images in various planes (axial, sagittal, and coronal). This aids in precise anatomical localization and facilitates a comprehensive examination of medical images. Moreover,

[vii] https://code.visualstudio.com/; accessed April 2024

MITK includes color maps and 3D image volume visualization features for a better comprehensive visualization of complex anatomical structures.

Appropriate colormaps are essential for an accurate visualization of image data and support to quickly interpreted and easily understood essential aspects of the visualized data[100,101]. MITK provides a set of color maps including GrayScale, Inferno, HotIron, Jet, Magma, Plasma, Turbo[102], and Viridis. Multiple transparent versions exist, where low values are made transparent.

MITK supports the level window approach for image visualization. This involves adjusting the intensity range to enhance specific features within an image. It consists of selecting a window width to control the range of pixel values and a window level to determine the center of this range. By manipulating these parameters, radiologists can optimize the visibility of structures.

2.4.2 Image Segmentation

One of MITK's key strengths lies in its advanced image segmentation capabilities. Segmentation, the process of partitioning an image into meaningful regions, i.e. to specify anatomical structures or pathological anomalies. MITK offers a comprehensive set of tools for various segmentation approaches, including manual, semi-automatic, and automatic segmentation[103].

MITK excels in advanced interactive image segmentation, annotating images for anatomical or pathological analysis. It provides manual, semi-automatic, and automatic segmentation tools. Manual segmentation in MITK involves user interaction via an intuitive interface, allowing precise delineation of structures. This hands-on approach is ideal for intricate anatomical features. In semi-automatic segmentation, MITK combines user input with automated algorithms, reducing manual effort and is particularly useful for large datasets or repetitive tasks[104].

Automatic segmentation in MITK utilizes advanced algorithms like machine learning for analyzing and segmenting images without direct user intervention. Effective in diverse scenarios, it identifies pathological regions, classifies tissues, and extracts anatomical structures. State-of-the art Deep Learning (DL) methods like the TotalSegmentator[105] and nnU-Net[106] were recently added to MITK's GUI.

2.4.3 Image Registration

MITK facilitates image registration, a crucial process in aligning and comparing different medical images. This capability is essential in various medical scenarios, including image-guided surgery, treatment planning, and monitoring disease progression. Image registration is realized using the MatchPoint[107] registration module of MITK.

2.4.4 Modular Developer and Application Framework

The modular developer framework lies at the core of MITK, playing a pivotal role in enhancing the toolkit's flexibility and extensibility. This architecture empowers developers to craft custom applications tailored to specific use cases and requirements. MITK's module system organizes functionality into discrete modules, each addressing a specific aspect of medical image processing, thereby enhancing code maintainability and scalability while facilitating the seamless integration of new features.

The MITK can be separated in two parts. A developer framework, that is called MITK and an interactive desktop application called MitkWorkbench. The developer

framework is written in C++ and depends on several open libraries as illustrated in figure 2.7.

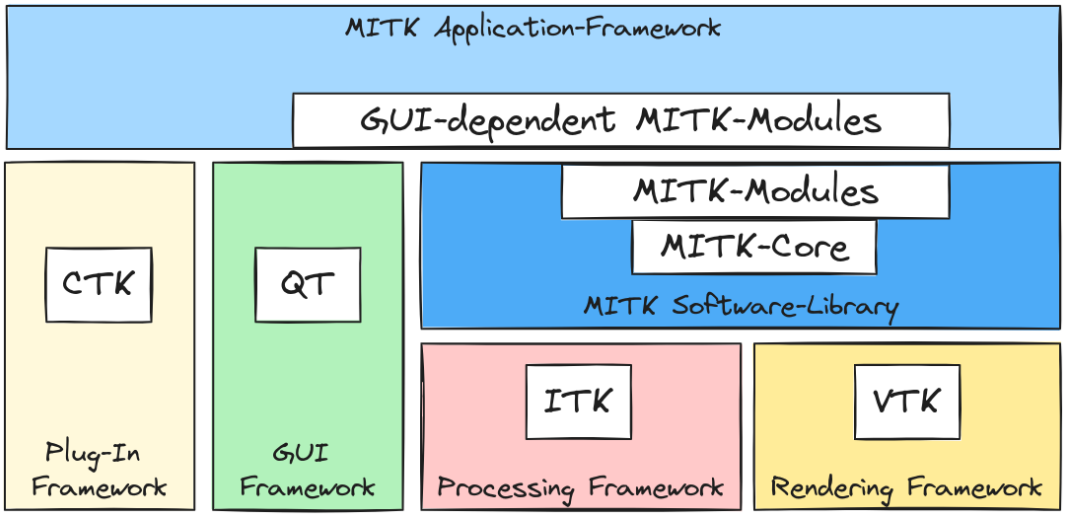

FIGURE 2.7. MITK uses the following libraries. The Insight Toolkit, which provides registration and segmentation algorithms, but is not designed for visualization or interaction[87]. The Visualization Toolkit, which provides powerful visualization capabilities and low-level support for interaction such as picking methods, rotation, movement and scaling of objects[108]. The Common Toolkit, which focuses on Digital Imaging and Communications in Medicine support and a plug-in framework[109]. The Qt Cross-platform application and User-Interface (UI) framework (Qt) as a framework for User-Interface (UI) and application support[110].These are the main libraries MITK is based on. For more functionality you can optionally include other libraries as well.

The MitkWorkbench is an interactive desktop application supporting a wide range of default image processing tools. The application is based on the above mentioned MITK developer framework. The user interface is based on a OSGi-inspired plugin framework, that is part of CTK. Open Services Gateway initiative (OSGi) stands for Open Services Gateway initiative, which is a specification defining a dynamic plugin system for Java[111]. The whole user interface is realized with those plugins - which were called "views" in the MITK context. The default user interface of the MitkWorkbench is illustrated in figure 2.8

MITK Superbuild The Superbuild automates the process of downloading, configuring, and building all the dependencies required by MITK, making it easier for developers to set up and compile the MITK framework on various platforms. By encapsulating all dependencies and configurations within a single build system, the Superbuild simplifies the setup and maintenance of the MITK project, ensuring that all required components are correctly built and integrated for efficient development and deployment of medical imaging applications.

MITK ProjectTemplate The MITK ProjectTemplate is a predefined structure for creating new projects within MITK[viii]. It provides a starting point for developers to create customized medical image processing or analysis applications using MITK. The template typically includes necessary directory structures and basic code examples to help developers get started quickly. By using the MITK ProjectTemplate, developers

[viii] https://github.com/m2aia/MITK-ProjectTemplate; accessed April 2024

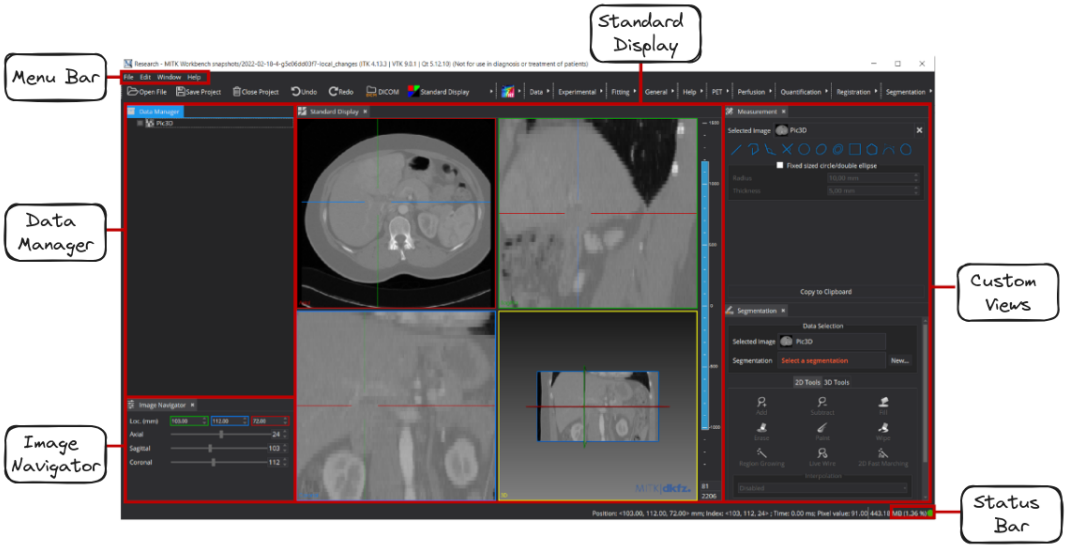

FIGURE 2.8. The MitkWorkbench's default user interface is a composition of multiple views, the menu bar (on the top) and the status bar (on the bottom). Main views required in each MITK based application are the DataManager and ImageNavigator. Custom views can be enabled, e.g. including segmentation and measurements. The standard display is a custom view tailored for image rendering for multi planar reconstructions of volumetric image data. A clinical CT image is shown in the standard display.

can ensure consistency in project structure, utilize best practices, and leverage common functionality provided by MITK, thereby accelerating the development process and enabling easy integration with the MITK Superbuild.

2.4.5 Base Image Class

In MITK, images are managed using the mitk::Image data structure. This class not only encapsulates the actual image data but also integrates information about the image's geometry. The pixel data within MITK images are stored as a contiguous memory block, supporting various data types.

The geometry of an image in MITK is defined by key spatial characteristics, including the origin, spacing, orientation, and size of the image. These properties are essential for correctly interpreting the image within a physical space context. The mitk::BaseGeometry class plays a crucial role here, as it houses the transformation matrix responsible for mapping pixel indices to physical coordinates, thereby bridging the gap between digital image data and real-world measurements. This integration of image data and geometry is fundamental to the robust processing and analysis capabilities in MITK.

- Origin: It defines the location of the first voxel in physical space.
- Spacing: It represents the physical distance between adjacent voxels in each dimension.
- Orientation: It describes how the image is oriented relative to the coordinate system (e.g., patient orientation in medical images).
- Size: It specifies the number of voxels along each dimension.

The integration of image data and image geometry in MITK forms a comprehensive representation of the image, crucial for enabling a wide range of visualization and analysis operations. MITK provides an extensive array of functionalities that allow for the manipulation of both image data and geometry. These capabilities are particularly vital in the field of medical image processing, supporting critical tasks such as image registration, which aligns multiple images into a single coordinate system, segmentation, which isolates specific anatomical structures for detailed study, and quantitative analysis, which extracts numerical data from images.

2.5 Containerization with Docker

Containerization enables the wrapping of a wide range of applications, providing developers with a seamless and efficient way for packaging, distributing, and running their applications. By encapsulating applications and their dependencies into lightweight, portable containers, Docker enables consistency across different environments. This approach improves the deployment process, reduces compatibility issues, and enhances scalability. With Docker's robust features such as version control, networking, and orchestration capabilities, developers can manage complex application architectures while ensuring flexibility and reliability. Some alternatives like Podman[ix] or LinuxContainers (LXC)[x] are available but rarely used.

Docker is an open-source platform that automates the deployment of applications within lightweight containers[112]. It provides a way to package and distribute applications and their dependencies as portable, self-sufficient containers that can run on any system supporting Docker. Docker utilizes operating-system-level virtualization to enable isolated environments for applications without the overhead of traditional virtual machines. Overall, Docker containerization empowers also biomedical scientists and developers to deliver applications faster, iterate efficiently, and maintain consistency across diverse deployment environments[2,4,113].

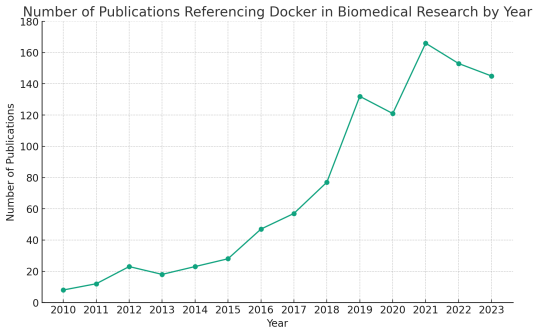

FIGURE 2.9. The plot above illustrates the number of publications referencing Docker in biomedical research by year, as recorded in the data. This visual representation clearly shows the presence and possibly growing interest in using Docker within the field over the time period 2010-2023[114].

Docker has become an increasingly valuable tool in biomedical research, offering a range of benefits that facilitate the development, testing, and deployment of medical software applications and research environments. The illustration (Figure 2.9 *PubMed Search Results (Keywords: Docker)*) demonstrates a notable presence and possibly an increasing trend in the adoption of Docker in biomedical research, over the time

[ix] https://podman.io/
[x] https://linuxcontainers.org/

period 2010-2023[114]. The use of Docker in this field can be attributed to several factors of containerization technologies:

- Reproducibility and Consistency: Docker ensures that computational research and experiments can be reproduced across different computing environments without discrepancies, which is crucial in validating scientific findings.
- Collaboration and Sharing: Researchers can easily share their Docker containers that encapsulate all necessary code, data, and dependencies, facilitating collaborative research and peer reviews.
- Scalability and Resource Management: With Docker, biomedical researchers can efficiently scale their applications across multiple environments—from local machines to cloud systems—optimizing resource usage and processing times for large datasets, such as genomic data.

2.5.1 Docker Images

Docker images are the building blocks of containers. They contain everything needed to run an application, including the code, runtime libraries, and dependencies. Images are created using Dockerfiles, which are text files that specify the configuration and instructions for building the image. For example, to create a simple Docker image that runs a script written in Python, the Dockerfile starts with a `FROM` statment inheriting a base image, copies the script into the image, and specifies the `ENTRYPOINT` command to start the script.

```
# Sample Dockerfile
FROM python:3.8
COPY . /app # copy data from disk to image
WORKDIR /app # set current working directory
RUN pip install -r requirements.txt # install dependencies
ENTRYPOINT ["python", "app.py"]
```

2.5.2 Docker Containers

Docker containers are lightweight, portable, and isolated runtime environments that run instances of Docker images. Each container is an instance of an image and runs as a separate process with its own file system, network, and resources. Using volume mapping, a directory on the host machine can be linked to a directory in the container, allowing for persistent or shared data between the host and the container. For example, to run a container from the previously created image and map a host directory to the container, a container can be started using the Docker CLI with a volume argument. This setup allows the container to run in isolation while still interacting with specific areas of the host file system.

```
# Command to run a container from the image with volume mapping
docker run --name image-processing \
  -v /path/to/host/directory:/path/in/container image-processing
```

In this command, '-v *path-host*:*path-container*' specifies the volume mapping. The '*path-host*' is the path on the host machine that is linked, and '*path-container*' is the path inside the container where the host directory will be accessible. This is particularly useful for applications that need to preserve data between container restarts, or

for development environments where code on the host needs to be tested inside a container. Docker can map volumes as read-only to protect specific folders, '-v *path-host*:*path-container*:ro'.

2.5.3 Docker CLI

Docker provides a command-line interface (CLI) that allows users to interact with the Docker daemon and manage containers, images, volumes, networks, and other Docker objects. The CLI provides a set of intuitive commands for building, running, and managing Docker containers and images. For instance, the Docker CLI can be used to build an image from a Dockerfile, list all running containers, and inspect the details of a specific container.

```
# Example CLI commands
docker build -t my-image .
docker ps
docker inspect my-container
```

2.5.4 Docker Registry

A Docker registry[112] is a storage and content delivery system that holds named Docker images, available in different tagged versions. Users interact with a registry by using Docker push and pull commands to upload and download images. Here is a breakdown of its core functionalities:

Storage of Images: the registry stores Docker images, which are pre-built application environments that users can download and use. These images can include everything needed to run an application, such as the code, runtime, libraries, environment variables, and configuration files.

Version Control: each image in the registry can have multiple versions, each tagged with a unique identifier. This makes it easy to release and track different versions of an application, facilitate rollbacks, and support multiple versions simultaneously.

Access Control: Registries can control who has access to upload or download images, which is crucial for private or proprietary applications where access needs to be restricted.

There are public and private Docker registries. The Docker Hub[xi] is the default public registry that anyone can use to find and share container images. For private or internal distribution, organizations can host their own registries using third-party solutions like JFrog Artifactory[xii] or Github[xiii].

[xi] https://hub.docker.com/; accessed April 2024
[xii] https://jfrog.com/de/artifactory/; accessed April 2024
[xiii] https://ghcr.io; accessed April 2024

Chapter 3

Interactive Multi-Modal 2D/3D MSI Data Analysis

In this chapter, the contributions of this thesis to the research fields of hyperspectral 2D/3D MSI processing in multi-modal setups are presented. The chapter is organized as follows: an overview and scope of the application, including necessary considerations about the used software frameworks and the graphical user-interface, are introduced in section 3.1. The data handling concepts for MSI datasets and related data are introduced in section 3.2 and, based on these concepts, interactive image registration tasks including multi-modal image fusion, 3D image reconstruction, as well as the evaluation and correction of registration results are introduced in section 3.3. Concepts for the programming language independent integration of third-party image processing methods[i] into M²aia are introduced in section 3.4 as well as concepts for the advanced data access focusing MSI deep learning applications in section 3.5.

Parts of this chapter have been published in:

- *Cordes et al., "M²aia—Interactive, Fast, and Memory-Efficient Analysis of 2D and 3D Multi-Modal Mass Spectrometry Imaging Data." GigaScience (2021)*[115]
- *Cordes et al., "M²aia Extension for Accessible Annotation Creation and Annotation Transfer for Mass Spectrometry Imaging in Multi-Modal Setups." International Mass Spectrometry Conference (2022)*[116]
- *Cordes and Wolf, "MITK Docker: An Open, Language-Independent Interface for Integrating Image Processing Pipelines into MITK." 6TH Conference on Image-guided Interventions (2023)*[117]
- *Cordes et al. pyM2aia: Python Interface for Mass Spectrometry Imaging with Focus on Deep Learning. Bioinformatics (2024)*[118]

3.1 Overview

MSI datasets are multidimensional, capturing detailed molecular information across spatially resolved samples. Understanding the structure of these datasets is crucial for researchers who analyze the molecular composition of tissue sections, cells, and other biological samples. The Mass Spectrometry Imaging (MSI) process is introduced in detail in section 2.1 *Mass Spectrometry Imaging*. Briefly, the data collection process involves a raster scan of the sample surface, which systematically covers the entire area and gathers data at each point. This approach creates a two-dimensional array

[i] The term 'third-party image processing methods' is used here to refer to command-line applications (scripted) that execute an image processing method and are not integrated directly into the code base of an interactive application.

where each pixel represents a unique location on the sample. Associated with each pixel is a mass spectrum that captures the relative abundance of ions for a given range of mass-to-charge ratios (m/z). The increasing lateral resolutions of the acquisition devices pose challenges in handling large MSI datasets.

Working with MSI data requires significant interactivity at various stages. Researchers need to select and control preprocessing steps, inspect spectra, and generate ion images. Interactive tools are essential for these tasks, enabling users to explore data and adjust parameters, based on responsive image visualizations.

Multi-modal MSI involves the integration of multiple MSI contrasts or the combination of MSI with other imaging modalities[6]. This approach is frequently used, especially for adding annotations from other modalities such as histology. Creating and working with annotations is yet another interactive task that underscores the necessity of interactivity in MSI data processing. The ability to correlate and analyze data from different sources enriches the interpretation of the biological context. In order to integrate multi-modal data, image-based registration techniques are employed, which spatially align image features in a common image space. Thus, these techniques facilitate the integration of corresponding anatomical and molecular features for integrated analysis.

Image registration is also critical for 3D reconstruction. As MSI is intrinsically a spatially 2D method, 3D reconstruction involves stacking separate 2D slices, which must be correctly aligned spatially[6]. To achieve accurate 3D reconstructions, image registration is used to align these slices properly. This step is similar to multi-modal registration, but is applied to reconstruct the spatial organization of the sample in three dimensions.

Interactivity is crucial for image registration, at least for visual inspection and utilization of the results, and often for initialization as well. Interactive tools allow users to manually adjust alignments, place control points, and evaluate registration accuracy. These tools are especially useful in challenging cases where automated methods may struggle, providing flexibility and precision in the registration process.

In summary, it can be concluded that interactivity is central to working with MSI, especially for multi-modal 2D/3D MSI. Existing open-source applications (see subsection 2.3.2 *Mass Spectroscopy Imaging Applications*) are limited in their capabilities to process multi-modal data or struggle to simultaneously process multiple MSI datasets, especially MSI datasets with different m/z ranges, and do not provide comprehensive 3D visualization features, such as multi-planar reformations, volume rendering and surface reconstruction, which are essential for 3D MSI.

The objective of this work is to introduce concepts for the advanced interactive processing of multi-modal 2D/3D MSI dataset, ultimately to support the creation of MSI-based deep learning applications.

Based on a review of existing open-access MSI applications (see subsection 2.3.2 *Mass Spectroscopy Imaging Applications*), the following research questions will be addressed: (1) Can an application framework be provided that allows researchers comprehensive interactive access to MSI datasets, simultaneously for multi-modal and 3D MSI? (2) Can this framework be used to implement the lacking capabilities of advanced interactive concepts for multi-modal image fusion and 3D image reconstructions? (3) Is it possible to use this framework for common workflows in the field of MSI, such as a rapid data exploration and creation of spatial annotations? (4) Is it possible to facilitate the integration and development of the more advanced processes of the molecular analysis workflow, such as the identification of relevant peaks and biomarkers?

The basic idea behind an interactive application for processing of multi-modal 2D/3D MSI is summarized within the illustration Figure 3.1 *Key Aspects of an Interactive Research Application for Multi-Modal 2D/3D MSI*.

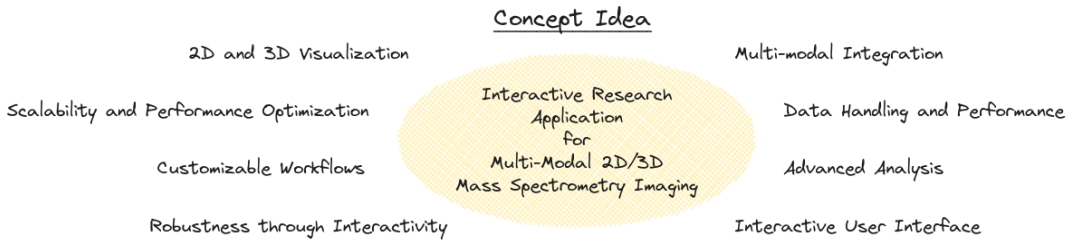

FIGURE 3.1. Key aspects of the concept for an interactive research application designed for multi-modal 2D/3D MSI. The application should facilitates dynamic exploration and analysis of both two-dimensional and three-dimensional MSI data, allowing researchers to intuitively navigate through complex biological samples. Enhanced visualization tools and analytical functionalities should be integrated to support comprehensive multi-modal investigations.

3.1.1 Application Scope

The primary objective is to provide an interactive and user-friendly interface that facilitates the manipulation and exploration of both multi-modal and 3D MSI data. Ensuring compatibility with various biomedical image data formats, such as those used in MSI or microscopy, is crucial for seamless data integration and analysis in multi-modal MSI. Furthermore, the application must be capable of fast and memory-efficient data processing and analytical tools, in order to accommodate the increasing size of biomedical datasets and to guarantee the responsiveness of visualizations, effective processing, and scalability of the application. The provision of corresponding capabilities would enable the development of process-intensive and interactive applications, such as those expected in the context of 3D reconstructions.

The functional scope of the conceptualized application is illustrated in Figure 3.2 *Scope of the Application*. Data Import/Export as well as the Hyperspectral Data Processing are introduced in section 3.2 *Concepts for MSI Data Processing*. Concepts for Image Registration are introduced in section 3.3 *Concepts for Image-based Registration in MSI*. Additionally, the Integration of Third-party Image Processing Methods is part of the concepts introduced in section 3.4 *Concepts for Integration of Third-Party Image Processing Methods*. Finally, the concepts for MSI-based deep learning are introduced in section 3.5 *Concepts for Deep Learning on MSI data*.

3.1.2 Architecture

In order to determine suitable software components to realize the previously mentioned required capabilities, interactive frameworks for 3D image processing were examined, and here primarily interactive open-source toolkits typically used in medical image processing of clinical images, such as CT or MRI. This approach was chosen because it promises long-term support for the visualization and manipulation of 3D image data. It is also expected that the use of 3D MSI will increase in the future. Most openly accessible interactive MSI applications focus on the analysis of 2D images, since the majority of MSI experiments are still designed to be purely 2D. Open-source application frameworks that are used in the clinical field and were considered in the course of the work include 3D Slicer and the Medical Imaging Interaction Toolkit (MITK). However, these toolkits have often limited support for biomedical images

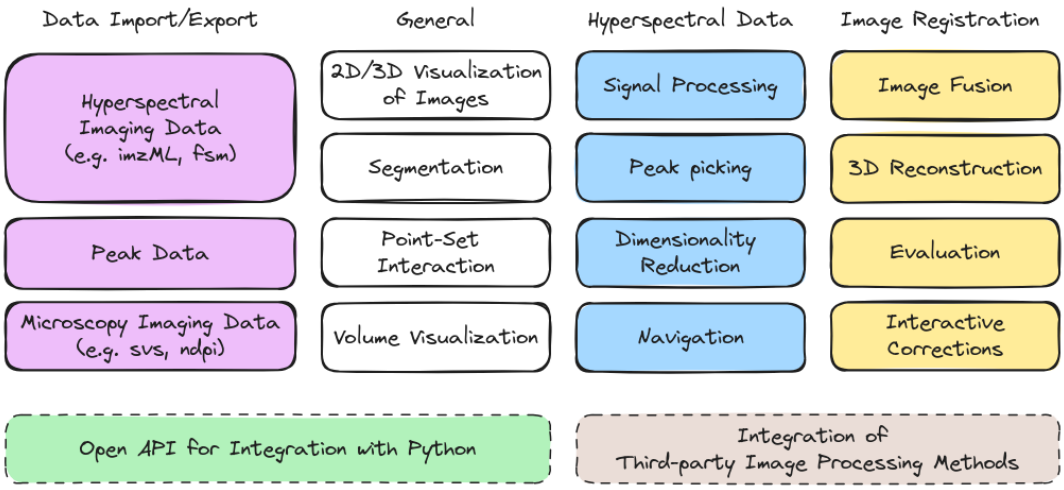

FIGURE 3.2. Scope of an interactive application that can handle hyperspectral and multi-modal imaging data in 2D and 3D. The boxes indicate individual features which are pursued in this work. The boxes with dashed lines indicate additional concepts that may be employed to facilitate the development of deep learning applications.

like high resolution microscopy images or the hyperspectral datasets generated by MSI. After careful consideration and based on the author's experience, the C++ application framework MITK[i] is the candidate of choice. MITK has a long history in the research and clinical image processing communities and has proven its abilities in various clinical disciplines[92,93,107,119,120]. It is maintained by the German Cancer Research Center (DKFZ) in Heidelberg as an open source project under a 3-Clause-BSD license. For a clean separation for the development of the MITK-based application, the MITK-ProjectTemplate[ii] provides a reliable starting point. The implementation of a MITK project template is then called an extension that can be easily integrated in the build process of the MITK superbuild. The MITK developer framework is described in more details in the background section 2.4 *Medical Imaging Interaction Toolkit*. The MITK framework provides a completely open-source code base built on top of ITK[87iii], VTK[108iv], and Qt[110v]. The working name for the extension is *MSI Applications for Interactive Analysis in MITK (M²aia)*[115].

The following paragraphs provide an overview of the individual topics illustrated in Figure 3.2 *Scope of the Application*. From left to right the topics Data Import/Export, General, Hyperspectral Data, and Image Registration are discussed one after the other.

Data Import/Export: in order to gain access to hyperspectral datasets, image file format support for MSI and FTIR datasets is required. Two data access strategies are considered:

1. Import entire data into the computer memory to support an instantaneously access to the spectral data. This is possible if the amount of spectral data is manageable, otherwise memory-limitations can become a problem.

[i] https://www.mitk.org/
[ii] https://github.com/MITK/MITK-ProjectTemplate
[iii] https://itk.org/
[iv] https://vtk.org/
[v] https://www.qt.io/

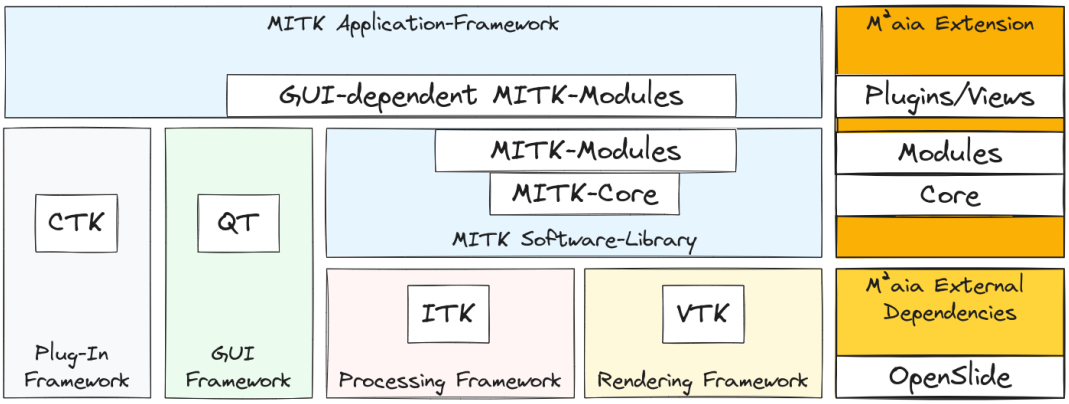

FIGURE 3.3. Architecture of the interactive application for processing hyperspectral and multi-modal 2D/3D imaging datasets. The application is based on the MITK framework and provides a set of modules and plugins for data import/export, data processing, image registration, and visualization. The modules provides data structures and import/export functionalities for the hyperspectral images and processing. The plugins provide interactive utilities for data import and image-based registration methods including multi-modal image fusion and 3D image reconstruction.

2. Import parts of the data using lazy-loading, a strategy in which data is only read from the hard disk when it is actually requested by the application at runtime.

The main challenge by working with MSI datasets is their potential huge data sizes[121]. To reduce the required amount of memory, a lazy-loading strategy is conceptualized for large hyperspectral datasets as those generated by MSI in subsection 3.2.3 *Ion Image Generation*. This is used to access spectral raw data on disk. The principle idea is to use the power of current computing capabilities, highly optimized code and the powerful read rates from solid state drives to omit to load entire datasets or to create interim data formats, as used by others[3,63]. Therefore, providing access to MSI datasets in ImzML format without interim conversions are of particular focus of this work[53]. It is an open-standard file format and is commonly used in open science projects, MSI applications and online databases like METASPACE[84] or Galaxy[76]. Optimized import routines for large imzML datasets is described in subsection 3.2.1 *Hyperspectral data import*. For the much smaller FTIR images in the proprietary but open file format FSM (PerkinElmer Inc.) the data import is realized by loading the full spectral dataset into the computer's memory.

In order to provide data for subsequent statistical analysis applications, it is required to export image data related artifacts. This includes images of all kind and image related data as peak lists or point sets. The native export utilities of MITK for gray value 2D/3D images is already supported by ITK's file writers. These include support for a wide range of standard file formats like jpg and png, but also file formats used in the medical image processing domain like NRRD[i] and Nifti[ii]. Point sets are used for image registration evaluation and in multi metric image-based registration approaches. Here, MITK's point set import and export is used. Peak lists support is provided by readers for import and exported of Comma-Separated Values (CSV) files.

The handling of high resolution image data generated by microscopy modalities including H&E stained whole slide imaging is challenging, due to to immense number

[i] https://teem.sourceforge.net/nrrd/index.html
[ii] https://nifti.nimh.nih.gov/

of pixels generated. The image dimensions quickly reaches the maximum texture size of a system (e.g. 8192x8192 pixels) and tiling strategies are required. However, due to the relative low spatial resolution of MSI compared to microscopy imaging, a representation of these images in a lower resolution could be sufficient for such a workflow. The integration of such images is described in subsection 3.2.6 *Data import of whole slide images*.

General Image Interaction: the aim of M²aia includes to process 3D and multiple images, possibly from different multi-modal sources. For the interactive implementation of interactive applications, such as 3D image reconstructions or multi-modal image fusion, appropriate visualization and interaction options must therefore be provided. By choosing MITK, which originates from the clinical field, corresponding functionalities can also be adopted for handling biomedical data. Additionally, with the advanced interaction and visualization concepts of MITK, the set of interactive tools provided by MITK itself can be used to support the desired workflows. Tools like interactive semi-automatic segmentation creation and interaction with point sets are inherited tools of the system and will be integrated within MSI related workflows (compare subsection 3.3.2 *Multi-modal Transfer of Image Annotations* and subsection 3.3.4 *Evaluation and interactive correction*).

Hyperspectral Data: for the processing of hyperspectral data generated by MSI, signal processing is an important part of the analysis workflow. It is required to reduce the influence of batch effects and includes processing steps like spectrum-wise normalization, smoothing, and baseline correction. This is described in more details in subsection 3.2.2 *Signal Processing*. Peak picking provides a first selection of molecular features and is described in subsection 3.2.4 *Peak picking*. Furthermore, dimensionality reduction methods are used in order to reduce the data volume while preserving structural features of the samples. The integration of dimensionality reduction methods is described in subsection 3.2.5 *Data Compression*.

Image Registration: the above mentioned capabilities of an interactive application for the processing of hyperspectral 2D/3D image data are targeting the main goals of providing concepts for interactive image-based registration scenarios. This interaction is necessary for a wide range of activities before and after an image registration task (subsection 3.3.1 *Multi-modal image fusion*; subsection 3.3.3 *3D Image Reconstruction*). The first inspection of unseen data, the selection of valid images, image content, or regions, and a potentially required manual alignment can benefit if full comprehensive interaction with the targeted biomedical datasets can be provided. Comprehensive interaction can be provided by user-based navigation capabilities and tools for manipulate image properties, such as position or orientation, and image-related data generation, such as the ability to generate point-sets. The visual evaluation of image-based registration results can also benefit from interactions with images, either for result overlays or for interactive correction strategies (subsection 3.3.4 *Evaluation and interactive correction*).

Integration of Third-Party Image Processing Methods: Third-party image processing methods are command-line applications (scripted) that execute an image processing method and are not integrated directly into the code base of an interactive application. The incorporation of image processing methods into MITK's C++ environment is a time-consuming process. Consequently, one objective of this research is

to develop concepts for integrating programming language-independent image processing methods into the interactive graphical user interface of M²aia. The concept is described in section 3.4 *Concepts for Integration of Third-Party Image Processing Methods*.

Integration of Third-Party Image Processing Methods:

3.1.3 Graphical user-interface

M²aia introduces Graphical User-Interface (GUI) elements to the MITK application framework that extends its functionality for analyzing a large number of MSI datasets and multi-modal biomedical datasets. These GUI elements are specifically tailored to support complex analytical tasks and provide the user with intuitive tools for efficient processing and interpretation of large datasets. The main goal of M²aia is to streamline the analysis workflow for researchers and clinicians working with MSI and multi-modal biomedical data. M²aia aims to simplify the process of data manipulation, visualization and interpretation and reduce the time and effort required for analysis tasks. It is designed to enable the analysis of a potentially large number of MSI datasets. The concept emphasizes scalability and flexibility to ensure that users can work seamlessly with different data sets while maintaining high performance and ease of use. The concept aims at three main goals:

- Custom data processing perspectives for a tailored arrangement of workflow-related views: Different researchers may have specific preferences or requirements for how they organize and visualize their data during analysis. Offering customizable perspectives allows users to arrange the User-Interface (UI) according to their specific workflows and analytical needs. This flexibility enhances user experience and facilitates more efficient data analysis. Essential views for the interactive navigation with MSI datasets are illustrated in Figure 3.4 *Spectrum Imaging Perspective*.

- Tailored views for the application of workflow-related methods: MSI data analysis often involves applying various computational methods and algorithms to process and interpret the data. A UI that provides tailored views for different analysis steps makes it easier for researchers to apply these methods effectively. For example, views optimized for signal processing, peak picking, data compression, and visualization can streamline the workflow and improve productivity.

- Intuitive and fast navigation for exploring MSI datasets: MSI datasets contain spatial information as well as mass spectra, which can be available in different forms depending on the type of pre-processing. For example, unprocessed spectra are provided as continuous-profile data, where each pixel contains an equal number of spectral intensities. In contrast, processed-centroid spectra represent a more complex scenario where each pixel contains a variable number of peak intensities. A well-designed user interface is essential to allow users to navigate through this data both intuitively and efficiently. This capability is crucial for researchers who want to examine the spatial distribution of molecules in a sample and pinpoint areas of interest for detailed analysis.

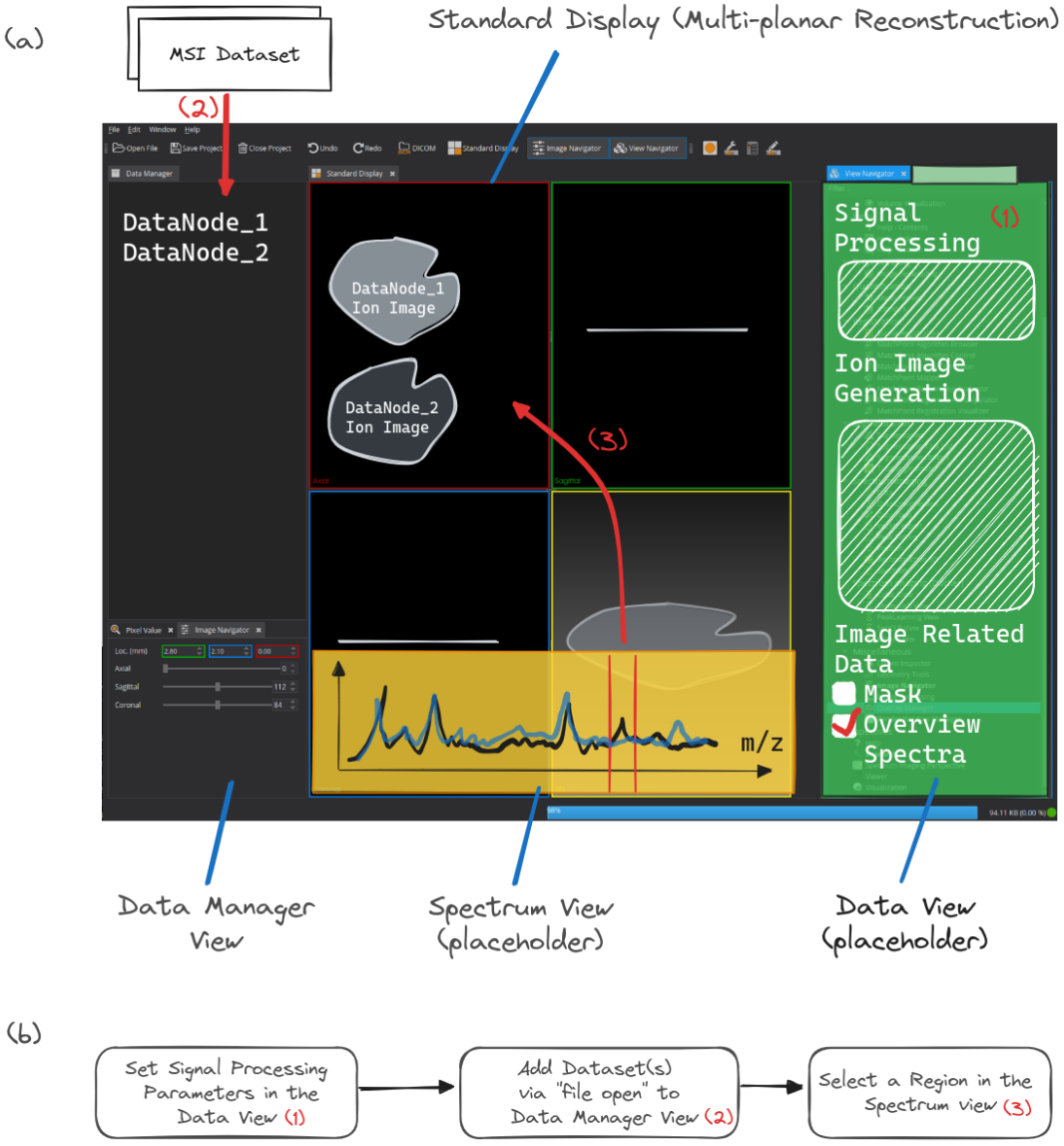

FIGURE 3.4. Proposed GUI layout: (a) Positioning of views on M2aia's default perspective are illustrated. The default MITK Workbench window is in the background. Placeholders for the Spectrum View (yellow), Data View (green) and two ion-image of MSI datasets are added as overlay. Data nodes in Data Manager View correspond to the ion-images. Two red horizontal lines highlight the latest selected range used for ion-image creation. The Spectrum View should be located at the bottom of the application in order to provides the user with overview spectra, such as the maximum spectrum and average spectrum, and interactive utils in terms of ion-image generation. Additionally, it should be able to show multiple overview spectra provided by multiple MSI datasets focusing on different molecule classes, such as lipids or peptides. The Data view should offer controls for defining signal processing methods and parameters for the ion-image generation process. Section (b) shows the first steps of loading MSI datasets. In step (1) the signal processing parameters are specified in the Data View; in (2) the MSI datasets are loaded; and in (3) the generation of ion-images can be triggered by interaction with the Spectrum View, e.g. by selecting m/z areas along the m/z axis.

3.2 Concepts for MSI Data Processing

This section introduces the concepts for the raw data import of hyperspectral images in subsection 3.2.1 *Hyperspectral data import*, as well as the lazy-loading strategy for parallel processing of hyperspectral data in subsection 3.2.3 *Ion Image Generation* and subsection 3.2.3 *Ion Image Generation*. With these strategies, the signal processing pipeline can be introduced in subsection 3.2.2 *Signal Processing*. The detection of peaks is described in subsection 3.2.4 *Peak picking*, which than can be used to apply data compression methods as described in subsection 3.2.5 *Data Compression*. The import of whole slide images of non-hyperspectral modalities is described in subsection 3.2.6 *Data import of whole slide images*.

3.2.1 Hyperspectral data import

Rapid response is key in user interactions. If interactive applications lag, user frustration increases, and conducting thorough analyses becomes a lengthy and expensive task. In scenarios like processing several large MSI datasets concurrently, for purposes such as 3D image reconstruction, massive data processing is inevitable. The most time-consuming phase is usually importing data. Presented here are optimized data import strategies aimed at accelerating the visualization of required elements in a collective, interactive image space.

Image Classes

To facilitate the integration of image data into MITK's interactive framework, compatible classes have been developed. The class hierarchy, focusing on image classes, is depicted in Figure 3.5 *Class diagram of handling hyperspectral images in M²aia*. At the core of this hierarchy lies the spectrum image base class[i], which inherits functionalities from the standard MITK image class. This inheritance allows leveraging MITK's extensive image processing capabilities, such as segmentation, measurements, and versatile visualization features in both 2D and 3D, including a diverse array of color maps and window-level adjustments. The class of spectral images serves as a proxy for hyperspectral datasets, which represent a single-channel image with spatial dimensions (width, height, depth) adapted to the MSI dataset. This proxy image can be used to store hyperspectral data processing results, primary for generated ion-images. In addition, the properties of the spectral images have been developed to optimize access to the spectral data. These properties include helper images such as the mask image for highlighting valid spectra, the index image for spatial representation of imzML dataset spectral indices and several normalization images with pixel-wise normalization constants for different normalization methods (subsection 3.2.2 *Signal Processing*). Furthermore, the provision of overview spectra, including mean and maximum spectra, is paramount for effective spectral navigation.

Two image classes inherit from the spectrum image class: the imzML spectrum image[ii] and the fsm spectrum image[iii]. The imzML spectrum image introduces properties designed for accessing spectral data stored in `.ibd` files. It employs a lazy-loading strategy (subsection 3.2.3 *Ion Image Generation*) to efficiently retrieve spectral data from disk. Through metadata parsing during data import (subsection 3.2.1 *Hyperspectral data import*), dedicated properties such as the file path to the `.ibd` file,

[i] Modules/M2aiaCore/include/m2SpectrumImage.h
[ii] Modules/M2aiaCore/include/m2ImzMLSpectrumImage.h
[iii] Modules/M2aiaCore/include/m2FSMSpectrumImage.h

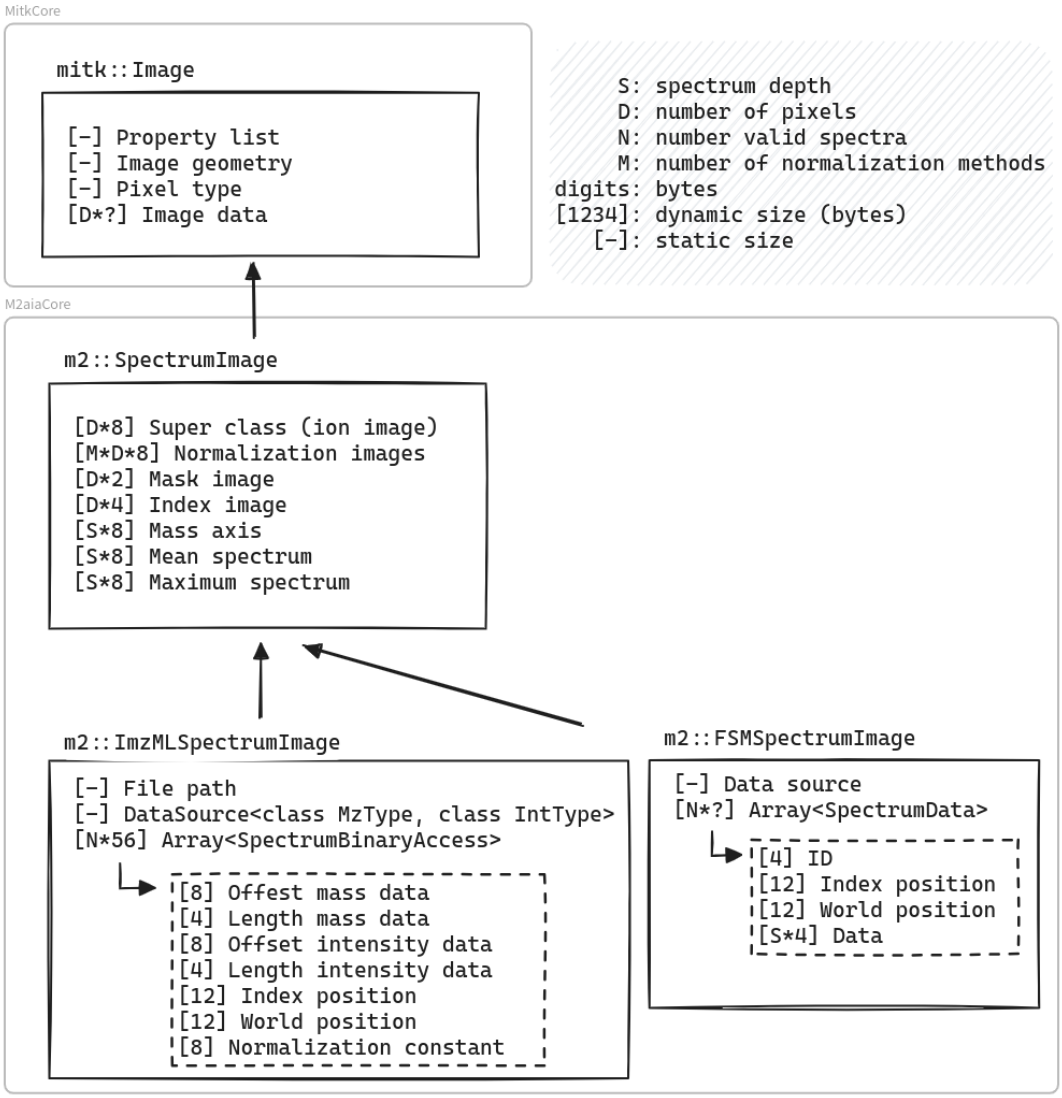

FIGURE 3.5. Class diagram of handling hyperspectral images in M²aia. Square brackets indicate the memory usage in bytes. Spectrum depth S ranges between 10^3 up to over 10^5. The number of pixels $D = width * height * depth$ in the MSI dataset. The number of valid spectra N ranges between 10^3 up to over 10^7. The number of normalization methods M, e.g. $M = |\{\text{TIC, maximum, sum, mean, RMS}\}|$ in M²aia. Boxes indicate classes. Dashed boxes indicate nested data structures.

a data-type generic data source, and information for accessing binary data for each spectrum are provided. In contrast, the fsm spectrum image does not employ a lazy-loading strategy. Instead, it loads the entire dataset into computer memory. This approach is suitable for smaller image sizes as typically generated by FTIR.

An essential aspect of the imzML spectrum image is its data source. This source facilitates access to `.ibd` intensity and mass data in a data type-independent manner. This is achieved through a templated class[i] capable of managing various data types for intensities and masses. The data type specification dictates the binary offsets and the interpretation of read bytes.

ImzML Metadata Parsing

The first step of each image analysis is the import of data into the application. One focus in this work is on optimizing the import of MSI data in format imzML, an open format for MSI datasets[53]. The image format is divided into two parts, which are stored in two separate files within the same folder: the file that describes imaging metadata uses the XML markup language (with `.imzML` file extension) and the file that provides the data in binary format (with `.ibd` file extension). In the further work imzML means the file format standard in entirety and a dot in front of `.imzML` or `.ibd` indicates one of the files of an imzML image. Here, the optimization of the metadata reader is particularly important in order to support the potentially large XML structures of the `.imzML` files, e.g. with regard to 3D MSI datasets that can contain huge number of pixels in a range from 10^3 up to over 10^7. Thus, `.imzML` files may contain several hundred thousand lines and XML elements. Typical XML readers now try to rebuild the hierarchical XML structures to represent these as an in memory object tree. These trees can then be used to query necessary imzML XML elements in order to access metadata image properties. These large `.imzML` files easily overwhelm some XML readers. To solve this problem, an optimized line-wise parsing for imzML XML elements is used. The final parsing procedure of the `.imzML` file is split into two steps. In the first step, all image related metadata elements are read from the `.imzML` file, which are used to describe the image dimension, pixel size, image position, as well as processing, device and acquisition details. In the second step, all spectrum-access metadata are parsed, providing access information to the data contained in the `.ibd` file. These are data offsets and data length information used to read spectrum-wise defined m/z-axis and corresponding intensity data.

The structure of `.imzML` files is hierarchical and consists of context elements and self-closing elements, which utilize associated controlled vocabularies (CV) to ensure consistency and interoperability in the data. Controlled vocabularies are a crucial part of the imzML format. They provide a standardized set of terms that can be used to describe the various aspects of the data and metadata. This standardization is key for ensuring that data from different sources can be compared or integrated effectively. CVs in imzML are used to ensure that terms used in the file are consistent and universally understandable.

[i] Modules/M2aiaCore/include/m2ImzMLSpectrumImageSource.hpp

```
...
<fileDescription>
    <fileContent>
        <cvParam cvRef="MS" accession="MS:1000294" name="mass spectrum" value=""/>
        <cvParam cvRef="MS" accession="MS:1000128" name="profile spectrum" value=""/>
        <cvParam cvRef="IMS" accession="IMS:1000030" name="continuous" value=""/>
        <cvParam cvRef="IMS" accession="IMS:1000080" name="universally unique identifier" value="643cf9e0
            -2151-4999-a2ba-b765ad18a296"/>
        <cvParam cvRef="IMS" accession="IMS:1000091" name="ibd SHA-1" value="78243
            ada1f9bcb34f7bedb982b5c59b126d53cf6"/>
    </fileContent>
</fileDescription>
...
```

LISTING 3.1. Exemplary context and CV elements in imzML. CV elements referes to different CVs called MS and IMS.

- Context Elements: These are the major sections of the file that define the overall framework. They include metadata about the experiment, such as the instrument used, the settings of the experiment, and the sample details. Context elements provide a structured way to store this information, making it easy for other researchers or software tools to understand and process the data.
  - FileDescription: Contains information about the `.imzML` file, including metadata like the software version and other descriptive elements.
  - ReferenceableParamGroupList: A container for groups of parameters that can be referenced multiple times throughout the document to reduce redundancy by grouping frequently repeated parameters and their values.
  - SampleList: Details about the samples analyzed, with each sample described by specific attributes that include information about the origin of the sample and any special treatments or preparations it underwent.
  - SoftwareList: Lists the software used for data collection, processing, or analysis, aiding in the reproducibility of experiments and the validation of results.
  - ScanSettingsList: Describes the specific settings of the scanning instruments or devices used for imaging, including aspects like scan speed, resolution, and other relevant technical parameters.
  - InstrumentConfigurationList: Provides information about the configuration of the mass spectrometry instruments used, including details about detectors, sources, and other critical hardware components.
  - DataProcessingList: Describes various data processing steps applied to transform raw data into the final form stored in the file, such as normalization, baseline correction, noise filtering, and other relevant processing steps.
- Self-Closing Elements: These elements are typically used to represent specific data points or attributes within the larger context elements. They are "self-closing" in XML terminology, meaning they do not contain nested elements but instead directly include data or references to data, often utilizing attributes to store the values. The self-closing CV elements in imzML, named 'cvParam', must provide multiple attributes:
  - cvRef: A CV reference that defines the attributes of this element.
  - accession: A unique identifier for the element as defined in the CV, which begins with 'MS' or 'IMS' followed by a seven-digit integer number, separated by a colon.

- name: The name of the element as specified in the CV.
- value: An attribute that remains empty for boolean-like flags; otherwise, it contains a value with a data type specified by the CV.

The values assigned to these attributes are governed by established controlled vocabularies specific to imzML[i] and .mzML[ii], saved in the OBO format 1.2[iii]. In listing 3.1 an exemplary imzML context and CV elements are shown.

```
// 'data' is a m2::ImzMLSpectrumImage *
using namespace std;
using FunctionType = std::function<void(const std::string &)>;
std::unordered_map<std::string, FunctionType> accession_map;
std::unordered_map<std::string, FunctionType> context_map;
auto stod = [](const string &s) -> double { return std::stod(s); };
std::ifstream f(data->GetImzMLDataPath(), std::ios_base::binary);
// shared variables
std::string line, context, tag, name, value, accession;

// This function 'ValueToProperty' uses a lambda expression to convert the attribute's value
// from a string to a specific type determined by the converter's return value. The converted
// value is then stored in the property list with that type. Lambda expressions provide a concise
// and flexible way to write inline functions that can capture variables from their enclosing scope.
//
// Parameters:
//    line - the string from which the value is extracted
//    converter - a lambda or function pointer that specifies how to convert the string value
//    default - an optional default value to use if the conversion fails or if the line is empty
//
// The map 'accession_map' uses lambda expressions to define how each specific attribute
// identified by its accession number (like "IMS:1000046" for pixel size x) should be handled.
// Each lambda is designed to pass its line of text to 'ValueToProperty', utilizing an appropriate
// conversion function ('stod' in these cases to convert string to double).
//
// Example lambda usage for attribute conversion:
const auto ValueToProperty = [&](const string &line, auto converter, const string &default = {}) {...}
accession_map["IMS:1000046"] = [&](auto line) { ValueToProperty(line, stod); }; // pixel size (x)
accession_map["IMS:1000047"] = [&](auto line) { ValueToProperty(line, stod); }; // pixel size y
// ...
// Example lambda usage for context attribute conversion:
context_map["software"] = [&](auto &line)
{
  attributeValue(line, "id", name);
  attributeValue(line, "version", value);
  context = name + " " + value;
};
//...
while (std::getline(f, line))
    if(IsContextElement(line))
        UpdateContext(line, context_map);
    else if(IsAccessionElement(line))
        ParseAccession(line, accession_map);
```

LISTING 3.2. Exemplary code snippet for defining custom parsing rules of accession elements. Here shown for the pixel size with explicit conversion to a double type.

A line-wise parsing can be effectively implemented for both context and CV elements through a sequential search process. When a context element is encountered, the parser tracks the current line's context by pushing the element's name onto a stack. Conversely, exiting a context element triggers the removal of the top element from the stack. For CV elements that contain an accession attribute, these are typically parsed as strings and subsequently stored in a property list associated with the image.

The property list of an `mitk::Image`, which acts as a key-value store, can be accessed through the MITK user interface. Each property is added using a string key that combines the accession number with the name of the CV element. Additionally, custom parser functions can be defined to handle specific attributes of context elements or to apply special type conversion rules for attributes of CV elements. This customization allows for more tailored data handling, enhancing the flexibility and functionality of the parsing process.

[i] https://github.com/m2aia/imzML; accessed April 2024
[ii] https://github.com/m2aia/psi-ms-CV; accessed April 2024
[iii] https://obofoundry.org/; accessed April 2024

A parser functions is demonstrated in listing 3.2. Spectrum-wise access information is stored in a list of metadata objects in the spectrum image see listing 3.3.

```
using BinaryDataOffsetType = unsigned long long;
using BinaryDataLengthType = unsigned long;

// BinarySpectrumMetaData structure holds metadata for a single spectrum.
struct BinarySpectrumMetaData
{
  BinaryDataOffsetType mzOffset; // start of the mz values
  BinaryDataOffsetType intOffset; // start of the intensity values
  BinaryDataLengthType mzLength; // number of mz values
  BinaryDataLengthType intLength; // number of intensity values

  itk::Index<3> index; // x,y,z index coordinates
  struct{float x, y, z;} world; // x,y,z world position in mm
  m2::NormImagePixelType inFileNormalizationFactor = 1.0; // normalization factor in file
};
```

LISTING 3.3. Metadata structure for storing spectrum-wise access information.

Initialization

Based on the information parsed from the `.imzML` file, a spectrum image can be initialized. The initialization process is illustrated in Figure 3.6 *Image Initialization Strategy*. This image initialization aims to provide necessary data for signal processing, image access, and spectral navigation of MSI data. It involves the creation of several helper images and the generation of overview spectra.

Helper Images:

- Mask Image: This image provides labels indicating valid pixels (intensity values >= 1) and background pixels and can be modified using MITK's segmentation tools. It is used to mask the image area used for ion-image generation. Type: `mitk::LabelSetImage` with pixel type `unsigned short`.
- Index Image: The index image provides a visual representation of the indices of the spectra as they were captured sequentially by the MSI device, and is used to facilitate regional queries. Type: `mitk::Image` with pixel type `unsigned int`.
- Normalization Image: The normalization image provides normalization factors for each spectrum. Type: `mitk::Image` with pixel type `double`.
- Ion Image: The ion-image is used for visualization. Only the image geometry is initialized for the ion-image. Type: `m2::SpectrumImage` with pixel type `double`.

Continuous-Profile Overview Spectra:

- Mean Spectrum: This spectrum is computed by averaging the spectral data across all pixels or a defined subset of pixels. The mean spectrum provides a general representation of the typical spectral features present in the entire sample, useful for identifying common peaks or trends across the dataset. Type: `std::vector<double>`.
- Centroid Spectrum/Maximum Spectrum: This spectrum is derived by taking the maximum intensity value at each m/z across all spectra. It highlights the most intense ions detected in the sample, regardless of their spatial distribution, which is particularly useful for spotting dominant compounds or contaminants. Type: `std::vector<double>`.

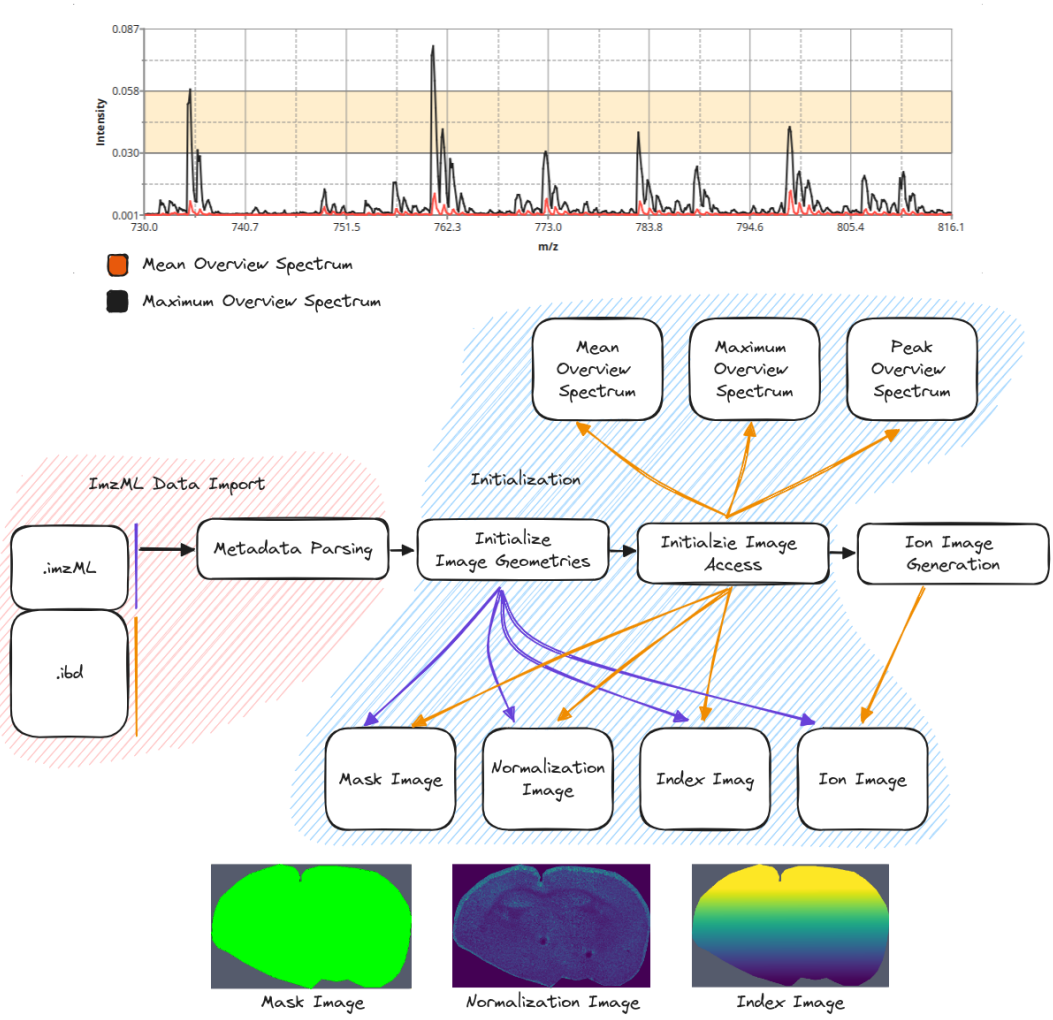

FIGURE 3.6. After the imzML data import and metadata parsing, image geometries are initialized for the mask image, normalization image, index image and ion-image. Subsequently, images are filled with data supporting the image access for subsequent processing.

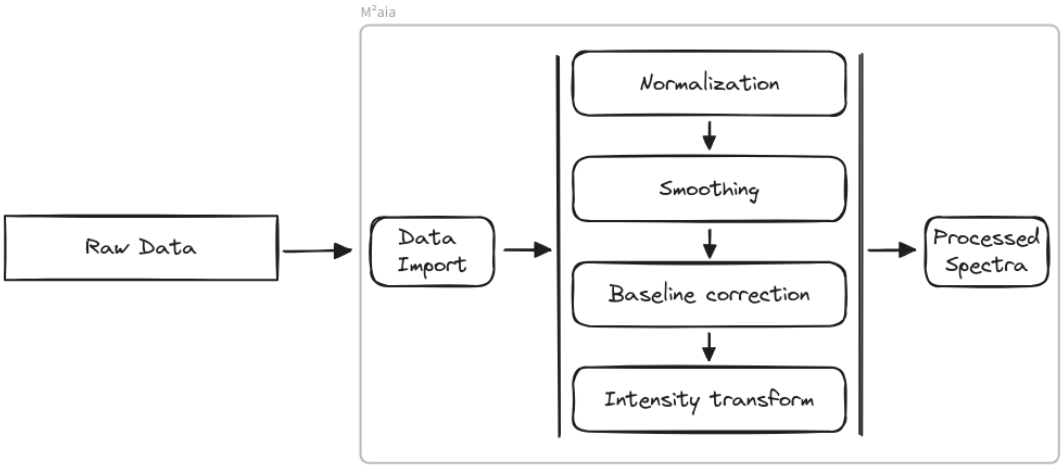

FIGURE 3.7. Schematic illustration of a MSI data analysis procedure. After data are accessible, signal processing steps are applied to the individual spectra.

Centroid Overview Spectra: each spectrum at each pixel is in centroid mode, meaning it only provides the peak locations (m/z) and intensities, not the full profile of the signal. Two different modes for centroid data in imzML file format exist:

- Continuous-Centroid: In this mode, each spectrum has the same m/z-axis, enabling direct calculation of the overview spectrum, and is derived by taking the average of intensity.
- Processed-Centroid: Since each spectrum can have different m/z-axis, direct summation is not feasible. In order to provide an overview spectrum for processed-centroid spectra, binning is applied by dividing the full m/z range into bins of equal size. The number of bins can be specified in the settings of M²aia. For each centroid spectrum, the intensities are mapped to the bins. For each bin the average intensity is calculated and is represented by its center m/z value. This provides a fast overview of processed-centroid spectra, but, depending on the size of the bins, it is possible that multiple peaks fall within a single bin, thus not every peak within the MSI dataset is reproduced exactly.

During the initialization of the image access, spectra are processed in parallel using the proposed map-reduce approach (see subsection 3.2.3 *Ion Image Generation*) under the utilization of the lazy-loading strategy (see subsection 3.2.3 *Ion Image Generation*). During accessing continuous-profile spectra, signal processing is applied (see subsection 3.2.2 *Signal Processing*). The result of this process is that the overview spectra and helper images are filled with appropriate values.

3.2.2 Signal Processing

Spectral analysis is a critical component of MSI as it can be influenced by various factors such as sample preparation, acquisition techniques, chemical interferences, fluctuating intensities due to matrix or surface non-uniformities, electronic variations and ionization phenomena[36,122]. Signal processing endeavors to mitigate these influences. Therefore, a signal processing workflow is implemented that includes normalization, noise reduction, baseline correction and intensity transformation methods. In Figure 3.7 *Signal Processing Procedure* a typical signal processing chain is illustrated.

The intensities of all spectra must be adjusted for better comparability (normalization). This is followed by variance stabilization and smoothing of the spectra

(smoothing). In order to minimize matrix effects and chemical contamination, the baseline of the spectrum is corrected (baseline correction).

Normalization/Calibration is used to compensate pixel-to-pixel intensity variations (see subsection 2.1.2 *Batch effects*). Intensity values of a spectrum S are normalized by

$$S(j) = S^{orig}(j) * f_S^{-1}, \tag{3.1}$$

where $S^{orig}(j)$ and $S(j)$ are the original and normalized intensities, respectively, f is the normalization value and j is the index related to the m/z position in a spectrum. The set of normalization strategies[18] include the total-ion-count (TIC) sum, mean, maximum, and root mean square (RMS). Calibration generally yields superior results compared to not using it[123]. The commonly employed TIC method is robust despite its simplicity[124], particularly in rectifying inequalities between technical replicates within the same measurement process.

Noise reduction/Smoothing Raw spectral data often contain various sources of noise, including electronic noise, background interference, and fluctuations unrelated to the analyte of interest. Spectrum-wise noise reduction using the Savitzky and Golay filter method[55] and Gaussian filter method, are implemented.

Baseline Baseline correction significantly improves the quality and clarity of spectral data by eliminating background noise and irrelevant signals. In Mass Spectrometry Imaging (MSI), raw spectra frequently include low-frequency signals unrelated to the target analytes. These baselines may originate from a variety of sources such as chemical noise, detector artifacts, or environmental influences. Without appropriate adjustment, these baselines can mask low-intensity peaks, distort quantitative analyses, and result in erroneous interpretations of the spatial distribution of molecules. The primary goal of baseline correction is to adjust the baseline towards zero, thereby enhancing the visibility and accuracy of spectral peaks. Ideally, the baseline correction should not influence the peaks height or the shape. As there is still no suitable method for measuring the baseline correction quality, an option is visual inspection[125]. Two common methods for achieving this are the running median filter and the TopHat transformation, illustrated in Figure 3.8 *Baseline Correction Example*.

The running median filter is a non-linear digital filtering technique exceptionally adept at reducing noise without substantially altering the signal's true peaks[126]. It operates by replacing each data point with the median of a predefined neighboring window that shifts continuously across the signal. The resulting running median signal is subtracted from the original to produce the baseline-corrected output. Similarly, the TopHat transformation employs morphological operations to emphasize small features and details in spectra that are smaller than the window size of the filter. This transformation is achieved by first eroding (applying a running minimum) and then dilating (applying a running maximum) the signal[127]. The resulting transformed signal is subtracted from the original to produce the baseline-corrected output.

Intensity Transformation: transformation of the intensity data facilitates the graphical representation and also serves to stabilize the variance. Intensities can be transformed by logarithmic transforms[129,130] or square root transform[131] to reduce the dynamic range of the data. These transformations are useful for visualizing and analyzing data with a wide range of intensities. It is applied after all other signal processing steps has taken place.

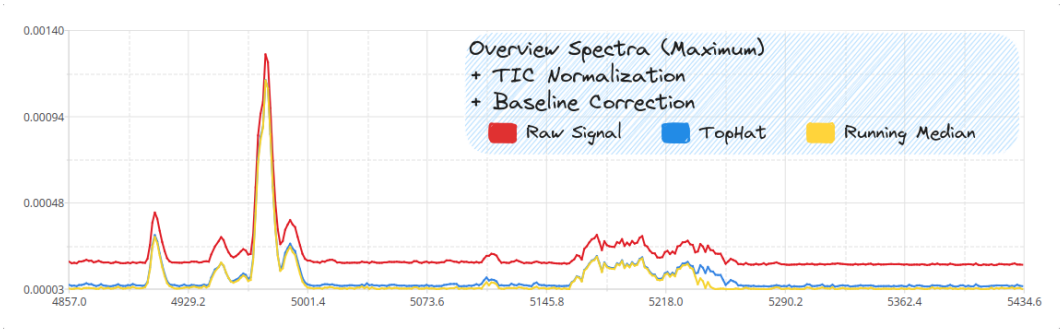

FIGURE 3.8. Exemplary, the raw signal (violet) is compared to baseline corrected signals: TopHat (red) and running median (blue). The half-window size is set to 50. Raw signal is derived from Slice 8 of the Peptide MALDI-TOF *Mouse Brain Dataset*[128] (see subsection 5.1.1 *Lipid/Peptide 3D APP NL-G-F Mouse Brain*) .

Sliding Window Approaches: baseline correction and noise reduction are implemented using a sliding window approach. The width of the sliding window is defined by the half-window size (hws) as $w = 2 * \text{hws}$. When using sliding window approaches, a common problem known as the 'border region problem' arises. This problem occurs due to the edges of the signal segments being processed differently from the interior regions. There are a few specific issues associated with the border region problem: At the borders of each window, incomplete data may be present, leading to inaccurate results. This is because the processing operation, which may involve convolution with a kernel or applying a feature extraction algorithm, assumes that the data within the window is complete. When processing near the edges, this assumption is violated, leading to artifacts or errors in the output. Portions of the signal near the edges of the windows may not be fully captured in the analysis. This loss of information can lead to biased or incomplete results, particularly in applications where accurate representation of the entire signal is important. Handling the border regions often requires special treatment, such as applying padding or using specific boundary conditions. Choosing the appropriate boundary conditions can be challenging and may depend on the specific characteristics of the signal and the processing operation being applied. The choice of window size can exacerbate the border region problem. If the window size is too small, the border regions become a significant portion of the analyzed signal, increasing the impact of edge effects. Conversely, if the window size is too large, it may obscure important spectral details in the signal. Addressing the border region problem typically involves careful consideration of the window size, choice of boundary conditions, and potentially applying techniques such as signal padding to mitigate the impact of edge effects. To handle this in M²aia, padding according to the window size is added to the signal including replicas of the first/last value of the spectral signal.

Order of Signal Processing Steps: effective signal processing is crucial for extracting meaningful data from MSI spectra. Key processing steps include smoothing, baseline correction, intensity transformation, and normalization. However, the order and implementation of these steps significantly influence the integrity and utility of the data.

1. To ensure the normalization factors reflects only instrument-introduced errors and not those introduced by data processing, it may be advantageous to calculate these immediately after data acquisition and before other preprocessing

steps. This preserves its integrity as a normalization standard.

2. Smoothing is typically applied after the normalization to reduce noise and enhance signal clarity, which aids in distinguishing peaks.

3. Baseline correction follows smoothing to eliminate background noise.

4. Intensity transformations are applied last.

3.2.3 Ion Image Generation

Generating ion-images using techniques like lazy-loading and map-reduce can significantly enhance the efficiency and speed of the process, particularly in scenarios involving large datasets or high-resolution images. Both strategies are described in the upcoming paragraphs within the context of image generation.

Lazy-loading Strategy: lazy-loading is a strategy primarily used to defer the loading of resources until they are actually needed. Loading tens of gigabytes of data into the computer's memory would be, at best, depending on the steps involved for signal processing, time consuming. To face this challenge, we follow the strategy of lazy-loading, resulting in minimal memory overhead during runtime. The lazy-loading is implemented for files in imzML format. The `.imzML` files contains all necessary information to define an image object in MITK with valid geometry (section 2.4 *Medical Imaging Interaction Toolkit*) and to define spectrum-wise metadata objects providing binary access information in the corresponding `.ibd` files (Figure 3.6 *Image Initialization Strategy*). Based on these information a lazy-loading mechanism can be implemented. As a lazy-loading strategy is intended to prevent large amounts of data from being held in memory, data read from the `.ibd` file is discarded immediately after processing. To further reduce the amount of data to be loaded, it is also possible in narrow applications to load only sub-ranges of the spectra.

Map-reduce Strategy: the map-reduce programming model is known to efficiently utilize available parallel computing resources by breaking down a large data processing task into smaller sub-tasks (mapping), distributing them across the parallel computing resources, and then combining the results (reducing) to produce the final output[132]. This strategy is applied to increase the performance of processing spectral data. Here, an equal number of spectra is mapped to individual threads, depending on the number threads used. If it is not possible to split without remainder, the remaining spectra are assigned to the last thread. Each thread processes the mapped spectral data and calculates desired intermediate results. Finally, the intermediate results are collected in the main thread and are reduced to the final results.

Generation Procedure: querying an ion-image requires to define an m/z value (center), a range around the center in Dalton (tolerance), the signal processing parameters, and the pooling strategy:

- Molecule of Interest: In order to generate a spatial distribution image for a molecule of interest, an associated m/z value x is required.

- Query Tolerance: The tolerance is used to take into account mass shifts caused by the imaging device. Typical values for the tolerance are associated with the peak width. Tolerance t can be a fixed value in Dalton or expressed as parts-per-million of the center value $t = 10^{-6}x$.

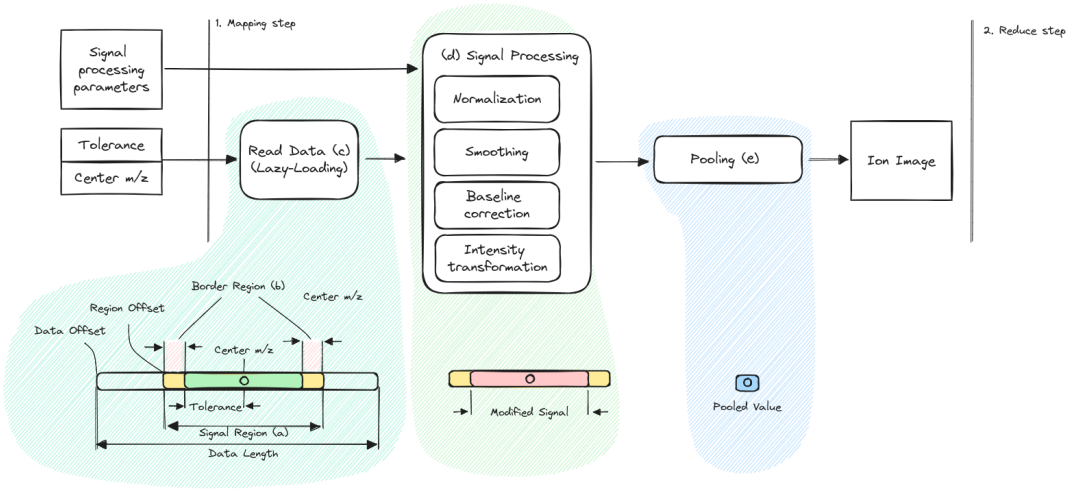

FIGURE 3.9. Illustration of spectral processing using map-reduce and lazy-loading. After a region of interest has been determined by the region offset and region length, data are read from disk and handed over to the signal processing. Subsequently, a pooling operation is applied to the modified signal. The final intensity value is returned and assigned as intensity to the spectrum-associated pixel position of the ion-image. Square boxes are data objects. Rounded boxes are processing steps.

- Signal Processing: Parameters and used methods of the signal processing pipeline are selected within the Data View.
- Pooling operations are used to reduce intensity values to a single value that can be used, e.g. for the generation of ion-image pixel data. Pooling operations are sum, mean, maximum and median for a given list of intensities.

User-actions within M²aia (see subsection 3.1.3 *Graphical user-interface*), can trigger the ion-image generation process:

- Data View: The Data view provides text input for center m/z value and tolerance.
- Center Selection: The center can interactively selected by double-click within the Spectrum View.
- Range Selection: The tolerance and center can interactively selected by drawing a rectangular selection in the Spectrum View. The m/z value of the molecule of interest is the mid-point and tolerance is the distance from center to the range border.

The process steps for the ion-image generation, illustrated in Figure 3.9 *Spectral data access during ion-image generation*, are:

1. Mapping Step: In the mapping step, an equal number of spectra is assigned to worker threads. Each of the worker threads processes the spectra assigned to it sequentially and performs the following steps:
   (a) Finding the Signal Region: Only a fraction of the intensity ranges are required for ion-image generation. These are typically around the m/z value of the molecule of interest and are defined by centers and the tolerance. In case of processed-centroid imzML data, where each spectrum has its own m/z-axis, the entire m/z-axis is read for each spectrum from disk in

order to be able to identify the positions of the required intensity values. Otherwise, for continuous-profile or continuous-centroid imzML data, the m/z-axis is queried only once.

(b) Determine Region Offsets: The region limits can be extended with additional values from the spectrum, in order to support signal processing operations, based on sliding-window approaches (subsection 3.2.2 *Signal Processing*).

(c) Lazy-loading: The intensities are read from the `.ibd` based on the identified intensity positions defined by the signal region and the region offsets.

(d) Signal Processing: The intensities loaded into memory are modified based on the defined signal processing settings.

(e) Pooling: A pooling operation is applied to reduced the modified intensities to a single value, such as the sum or average, in order write it to the spectrums pixel position in the ion-image.

2. Reduce Step: This step is skipped when generating ion-images, but can be relevant if the values of the mapping step are to be reduced, such as in creating overview spectra.

3.2.4 Peak picking

Peak picking involves identifying peaks within a spectrum, which provides valuable information about the m/z values and their corresponding intensities. In MSI, peaks are typically detected by locating local maxima that surpass a predefined noise threshold, often using a sliding window approach. This process ensures that only significant peaks are captured while minimizing the inclusion of noise. To determine this noise threshold, the median absolute deviation (MAD) of the spectrum intensities is commonly employed as an estimate. Moreover, the identification of monoisotopic peaks, which are the peaks corresponding to the most abundant isotopes of a given molecule, is enabled by automatic Poisson peak harvesting[133]. This method is used to detect and differentiate monoisotopic peaks from other peaks in the spectrum. Exemplary results of both strategies are shown in Figure 3.10 *Peak picking strategies*.

3.2.5 Data Compression

Data compression includes techniques for reducing data size. These are crucial for the visualization of high-dimensional hyperspectral data and provide a tool to make relationships in the data comprehensible. These techniques primarily include dimensionality reduction and clustering (see subsection 2.1.3 *Data Processing*).

Ion Images: even if it is a very basic type of dimensions reduction, the ion-image generation can also be seen as such. This approach simplifies the hyperspectral dataset into a 2D image, typically used to spatially visualize molecular distributions. This is described in detail, with focus on process optimization by parallelization, in subsection 3.2.3 *Ion Image Generation*.

Dimensionality Reduction: advanced unsupervised techniques for dimensionality reduction are used in visualizing hyperspectral data through spectral correlations depicted in (colored) multi-channel images. Techniques such as Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) are provided. PCA transforms correlated variables into a set of linearly uncorrelated

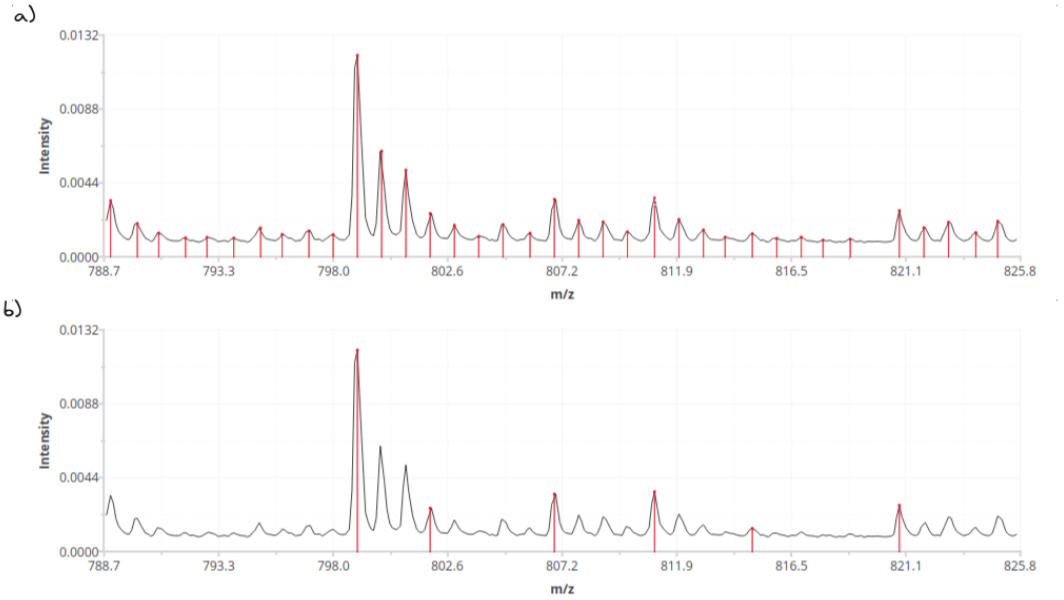

FIGURE 3.10. The mean overview spectrum is shown in both plots and is derived from Slice 5 of the Lipid MALDI-TOF *Mouse Brain Dataset*[128] (see subsection 5.1.1 *Lipid/Peptide 3D APP NL-G-F Mouse Brain*) . In plot (a), local maxima are marked with red dots and lines. In plot (b), only monoisotopic peaks are shown in.

variables known as principal components[17,134]. t-SNE is designed to generate low-dimensional visualizations that retain the local structure of high-dimensional data[135]. Typically, a preliminary PCA reduction is employed to prepare data for t-SNE, optimizing processing time. The PCA and t-SNE workflows in M²aia are illustrated in Figure 3.11 *Dimensionality Reduction Strategies*.

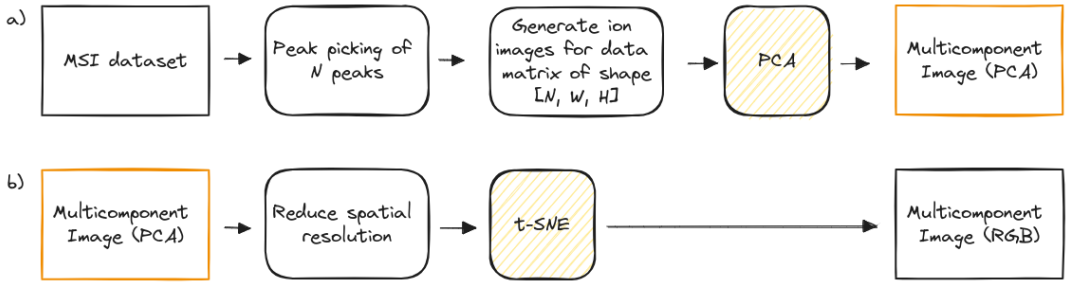

FIGURE 3.11. PCA and t-SNE dimensionality reduction strategies are illustrated. Strategy (a) uses the ion-image generation for multiple peaks to generate the required data matrix to apply a PCA. Strategy (c) uses the result of a PCA, which can be reduced in spatial resolution to speed-up the process, is used to apply the t-SNE in order to generate a three component image (RGB). Yellow shaded background highlight the dimensionality reduction method. Colored frames highlight the usage of the results of one approach in a subsequent approach.

Unsupervised Clustering: clustering methods, such as k-means, facilitate image segmentation by categorizing spatio-spectral data into discrete labels. K-means clustering assigns pixels to a predefined number of clusters, grouping them based on the spatial proximity of their mass spectra. Each pixel is assigned to the cluster with the closest mean, effectively segmenting the image into similar regions.

Details on Data Input for Data Compression Methods: in order to start a data compression method, a data matrix is prepared, based on a user-defined list of m/z values. This list can be generated interactively, using the peak picking utilities provided by M²aia (see subsection 3.2.4 *Peak picking*), or it can be loaded from disk. Ion-images are generated for each m/z value. The pixels of the ion-images are used to fill the data matrix. This has two advantages: first, it avoids problems with processed centroid data. Since an ion image is defined over a m/z range, even slight variations of the centroids in each spectrum can be compensated. Second, it is often not necessary and not always useful to use the full spectral depth of a data set, as the limits of memory requirements are quickly exceeded.

3.2.6 Data import of whole slide images

Importing whole slide images presents unique challenges due to their immense file sizes and high resolutions. Modern graphics hardware is limited by the maximum texture size it can handle, often making it impossible to render these images directly, as they can reach gigabytes in size with dimensions of tens of thousands of pixels. Consequently, it is impractical to display such large images as single textures. This limitation typically requires implementing strategies for on-demand loading and creating image pyramids. Image pyramids store multiple resolutions of the image, allowing for efficient zooming and panning by loading only the necessary resolution segments. Additionally, techniques like data streaming and caching are essential for managing memory effectively, ensuring that only needed parts of the image are retained in memory at any given time[136]. Major modifications in MITK's software architecture would be essential to incorporate these required features for visualizing whole slide images.

Nevertheless, the high spatial resolution differential between MSI datasets and whole slide images suggests that supporting the extreme resolutions of whole slide images might not offer significant benefits. Instead, selecting spatial resolutions that match critical features of the MSI datasets, such as outlines and structural characteristics of tissue sections, and that remain within manageable memory limits, are sufficient to support the workflows described in the upcoming section 3.3 *Concepts for Image-based Registration in MSI*. To accommodate a variety of whole slide image formats, the OpenSlides[i] library is employed, supporting file formats such as ndpi, svs, and tiff[137]. When corresponding files are loaded into M²aia, a dialog appears that offers the user a selection of internal image pyramid representations based on the file format. If the selected resolution exceeds the maximum texture size, the image will be represented by multiple images within MITK, each representing a portion of the image at the selected spatial resolution.

The typically colored images are composed of three channels for red, gree, and blue (RGB) or with a four'th alpha channel (RGBA). During the import of whole slide images the image data can be processed. The channels can be imported separated, picked individually, or processed in order to combine the three/four channels into one representative image. To combine RGB values, the luminance image method is used. It calculates a weighted combination of the RGB values in each pixel to create a gray-value representation. The luminance formula was used as it is defined in VTK[138,139].

[i] http://openslide.org; accessed April 2024

3.3 Concepts for Image-based Registration in MSI

In this section, the interactive concepts for the image registration workflows are introduced for multi-modal image fusion (subsection 3.3.1 *Multi-modal image fusion*) and the 3D image reconstruction (in subsection 3.3.3 *3D Image Reconstruction*). Furthermore, interactive concepts for the evaluation and correction of image-registration results are described in subsection 3.3.4 *Evaluation and interactive correction*.

3.3.1 Multi-modal image fusion

The integration of MSI data with images from a different modality can be used to gain deeper insights into biological samples. MSI offers spatial mapping of molecular species within a sample, providing information about the distribution of various compounds. Conversely, other imaging modalities may offer detailed structural and cellular context to complement MSI data. The fusion of MSI and other imaging modalities involves image-based registration, allowing for the visualization and analysis of molecular distributions within the context of cellular and tissue structures. This integration enables to correlate molecular signatures with specific cellular features, such as cell types, providing insights into biological processes, disease mechanisms, and drug responses.

In this context, the development of utilities for interactive fusion of multi-modal images becomes paramount. The interactive concepts aim to facilitate exploration and analysis of fused datasets, so that researchers can extract meaningful information. The image fusion procedure uses image-based registration methods to determine parameters of the desired transformation.

The key challenge in multi-modal image fusion with MSI is to find representative structural images for the hyperspectral datasets that can be used for image-based registration. There are two main strategies: In the first strategy, the MSI dataset is represented by a proxy image generated from the hyperspectral data itself. These images are bound to the relatively low spatial resolution of MSI devices and may not represent the whole structural complexity and features of a biological sample. Dimensionality reduction methods are used to generate, for example, an ion-image, a single principal component image, or another image using the applicable methods. In the second strategy, an optical image (typically used for MSI device calibration) is used to realize an indirect image fusion with MSI data. This requires, knowledge about the transformation from the optical image to the MSI image space, typically provided by the raw data output of MSI devices. The whole interactive image-based registration workflow for multi-modal image fusion with MSI is illustrated in Figure 3.12 *Image registration workflow*.

3.3.2 Multi-modal Transfer of Image Annotations

Parts of this subsection have been published in:

- *Cordes et al., "M²aia Extension for Accessible Annotation Creation and Annotation Transfer for Mass Spectrometry Imaging in Multi-Modal Setups." International Mass Spectrometry Conference (2022)*[116]

The primary challenge in multi-modal image registration lies in the inherent differences in image characteristics and the information each modality conveys. For example, MSI provides molecular information, while histological images offer structural details. Pathology and histology involve the microscopic examination of

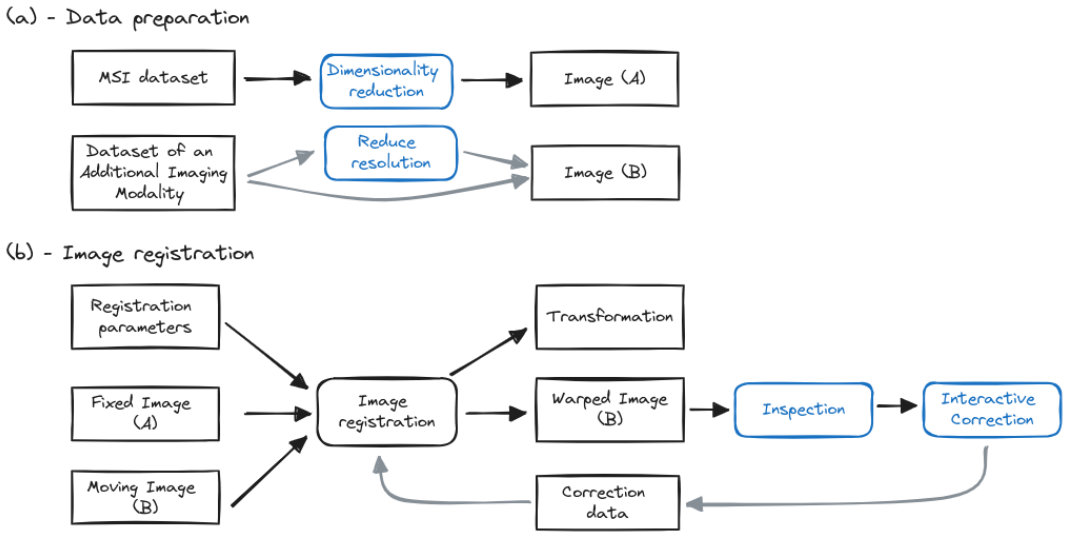

FIGURE 3.12. Illustration of the image registration with input and output data. Gray arrows show optional paths. Blue involves interactive steps within M²aia. A and B highlight the imaging modalities.

tissue samples to diagnose disease. These modalities typically use stains and dyes to highlight structures within tissues, providing crucial information on morphological and structural contexts[10].

The task is to align these images accurately to correlate molecular features with morphological and structures features. In order to transfer annotations between MSI and histology images, rigid, affine, or non-rigid transformations can be applied, adapting to the specific needs of the modalities involved. For the realization of the multi-modal image fusion in M²aia see the concepts as introduced in subsection 3.3.1 *Multi-modal image fusion*.

Once images are aligned, spatial annotations from one modality (e.g., regions of interest identified in histology images) can be transferred and superimposed onto the MSI images within the interactive framework of M²aia. This is realized by re-applying the same transformation parameters of the multi-modal image fusion to the annotated image. The only condition is that the annotated image and the image on which annotations have been made share a common location in the world-coordinate space (see section 2.4 *Medical Imaging Interaction Toolkit*).

Post-registration, the accuracy of the overlay and the transferred annotations are assessed through qualitative and quantitative means, ensuring that the molecular and structural correlations are accurately represented (see subsection 3.3.4 *Evaluation and interactive correction*).

This facilitates a comprehensive understanding of the spatial distribution of molecules in relation to the tissue structure and opens up new avenues in research and clinical applications, such as correlating histopathological findings with molecular imaging biomarkers, enhancing tumor margin assessments, and facilitating more precise and targeted therapies.

3.3.3 3D Image Reconstruction

Interactive 3D reconstruction is used to create a 3D MSI volume by aligning consecutive 2D MSI datasets to each other. This is realized by applying subsequent rigid and deformable image-based registration steps. The concept of the interactive 3D image

reconstruction is illustrated in Figure 3.13 *3D image reconstruction workflow*. Technical challenges are the provisioning of multiple MSI data at the same time, the choice of which image content is used for registration, and the parameterization of the rigid and deformable registration steps. The complete 3D image reconstruction process can be separated into three steps.

First (a) 2D MSI datasets are imported and due to the previous described capabilities of the optimized data import (subsection 3.2.1 *Hyperspectral data import*), a simultaneous and interactive access to a large number of 2D MSI datasets can be achieved. A dimensionality reduction method is used to reduce the hyperspectral data set into a 2D image, such as an ion-image or a principal component image. At this point it is important to select a structural-rich image content, that can be used for a image-based registration. These images are used for a first user-driven and interactive inspection of the image contents in M²aia.

3D image reconstruction often necessitates capturing and registering a substantial number of 2D MSI datasets. However, variations in sample preparation, matrix application (MALDI MSI), and instrumental settings can lead to inconsistencies across datasets. Such heterogeneity can complicate data integration and analysis during the reconstruction process. At this stage, users have several options to streamline the process:

- Organizing Images in a Matrix: Images can be systematically arranged in an image matrix. This arrangement aids in visualizing the sequence and spatial orientation of the images, facilitating easier manipulation and analysis.
- Excluding Images: It is possible to exclude certain images from the dataset, including calibration images or any images deemed unsuitable for registration. This step ensures that only relevant and high-quality images contribute to the 3D reconstruction.
- Manual Alignment: To improve the 3D reconstruction result, users can manually align images based on the content of neighboring images. This manual intervention allows to align image features across a series of images, in order to provide optimal start-conditions for the automatic image registration steps with elastix.
- Selection of Start Images for 3D Reconstruction: Users can select a specific image as starting point for 3D reconstruction. This selection is crucial as it determines the foundation upon which the entire 3D structure will be built.

In the second step (b), the selected and structural-rich images are used to find transformations between the images. To create a 3D MSI volume, consecutive slices are aligned to each other, applying subsequent rigid and deformable image-based registration steps in a fully automated way. At the beginning of the procedure the registration parameters are defined. In each iteration a fixed image is selected (manually in the first iteration of step (a)) and a neighbor image (e.g. the next image in ascending or descending order) is assumed to be the moving image. Before a registration is started, pre-registration transforms are applied to the selected fixed image (section 2.2 *Image-based Registration*; Figure 2.5 *Optimization process in image based registration*). This is illustrated in detail in Figure 3.14 *Registration steps for the 3D image reconstruction*.

In the final third step (c), a 3D ion-image is generated by using the pairwise transformations and 2D images of the MSI datasets. For each 2D image, the transformation

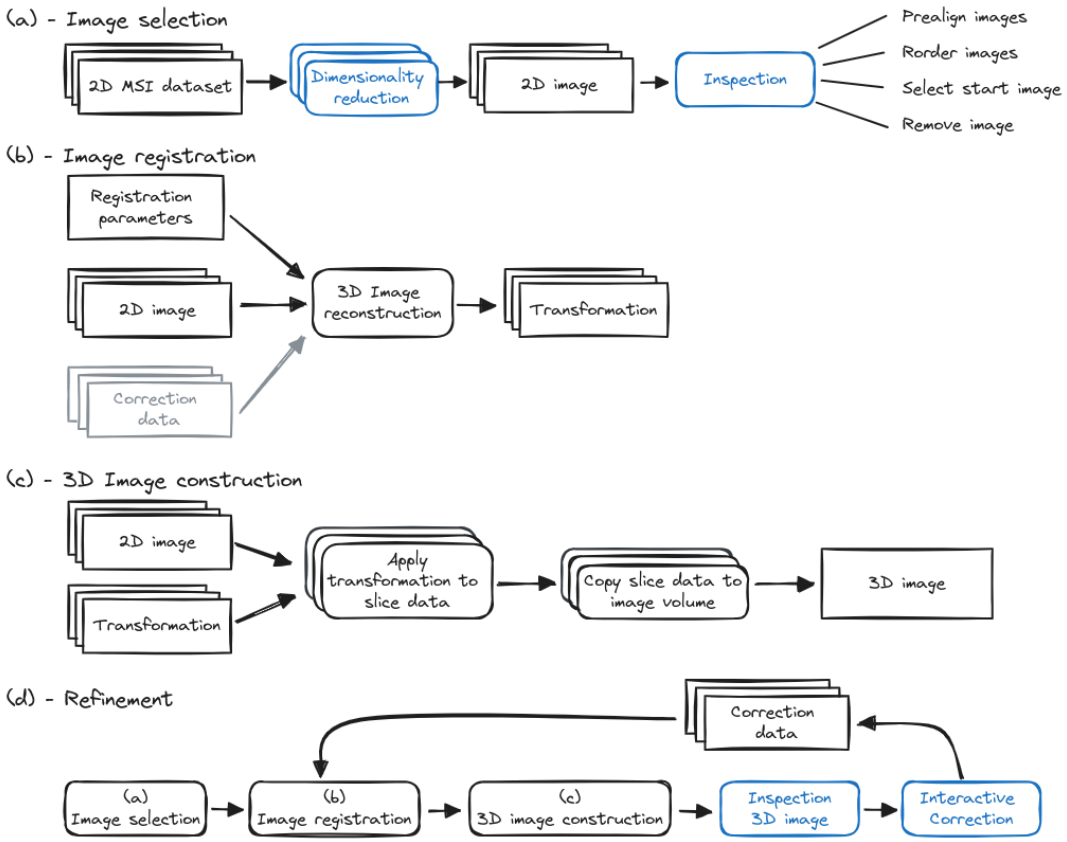

FIGURE 3.13. Three steps of the 3D image reconstruction workflow are illustrated. In (a), the preparation process is shown. In sub-illustration (b), the actual reconstruction is shown with respect to inputs and outputs. In (c) the 3D image is filled with data based on the results of step (a) and (b). In (d) an optional refinement workflow is illustrated. Gray shows that it is optional. Blue involves interactive steps. Stacks represent multiples of named objects.

is applied to warp the image content of the 2D images. The image content is copied slice-wise into a 3D dedicated image volume.

In cases when the results of individual registrations are not sufficient accurate, a refinement step (d) can be added to the workflow (subsection 3.3.4 *Evaluation and interactive correction*). This is realized by adding corresponding points in both modalities which are used for a multi metric registration approach that includes an image similarity metric (e.g. Mutual Information) in weighted combination with corresponding points Euclidean distance metric.

3.3.4 Evaluation and interactive correction

The registration results from the multi modal image fusion and the 3D image reconstruction are evaluated interactively by visualization and/or by calculation of the Target Registration Error (TRE). Visual inspection is based on MITK functionalities including the interactive visualization of 2D/3D images. For the visual evaluation of registration results, superimposed image visualizations are used. Available method include blending, checkerboard, color blending, difference and wiping. In 3D reconstructions, the volume is inspected slice by slice to detect misalignments between adjacent slices.

For quantitative evaluation, corresponding pairs of points can be set within the fixed and moving image using the point set interaction tools provided by MITK.

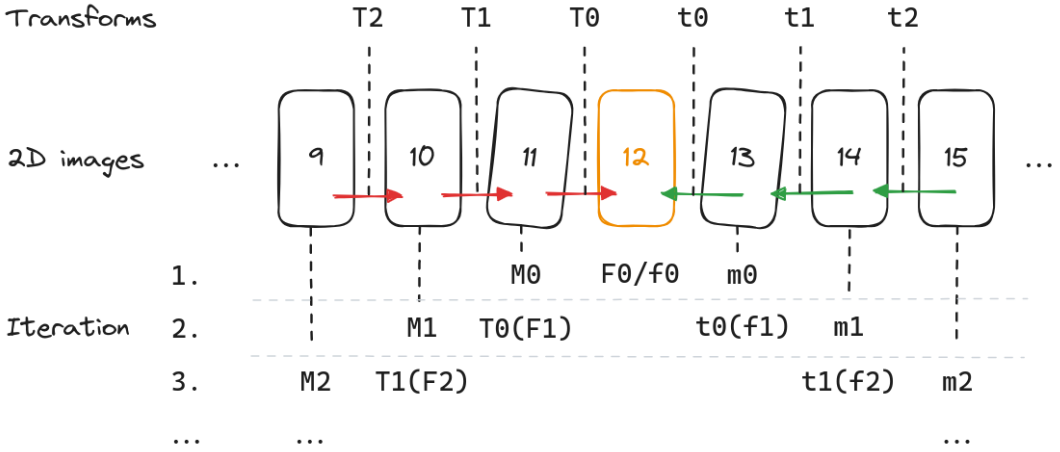

FIGURE 3.14. Registration steps for the 3D image reconstruction. Upper case letters indicate the downward branch of subsequent image registrations and lower case letters indicate the upward branch of subsequent image registrations. T/t are transformations, M/m are moving images, and F/f are fixed images.

Based on these point pairs, the TRE[140] can be calculated (Figure 5.5 *Target registration error (TRE)*).

3.3.5 Registration Backbone

The image registration is based on the elastix toolkit (subsection 2.3.6 *Frameworks for Image-Based Registration*). The elastix registration utilities are used for this process, providing access to appropriate transformation models and similarity metrics. For instance, mutual information is a common choice for multi-modal registration due to its effectiveness in measuring statistical dependence between datasets. Image registration utilities are realized within a separate MITK external project called *mitk-elastix*[i]. This external project is independent of MSI utilities of the M²aia project, creating a more general solution to image-based registration problems in MITK. This includes helper classes for developers and a view for image selection and parameter tuning. Default parameters for rigid and deformable parameters are chosen according to previous work[115].

The image-based registrations are realized with the help of the command-line tools provided by elastix, called `elastix` and `transformix`. For this purpose, images and parameter files are temporarily stored on disk in a dedicated workspace folder before a registration step or, if the data is already stored on disk, corresponding file system-links are provided.

3.4 Concepts for Integration of Third-Party Image Processing Methods

Parts of this section have been presented in:

- *Cordes and I. Wolf, "MITK Docker: An Open, Language-Independent Interface for Integrating Image Processing Pipelines into MITK." 6TH Conference on Image-Guided Interventions (2023)*[117]

[i] https://github.com/m2aia/mitk-elastix

3.4.1 Overview

Innovative data analysis techniques are revolutionizing numerous fields, with MSI proving to be a particularly dynamic area. MSI involves image data, the effective processing and analysis of which is crucial. The way researchers interact with data has a major impact on their ability to gain insights and make informed decisions.

However, novel methods are often distributed as executable scripts, posing significant challenges for researchers who are not proficient in scripting languages. They may struggle with the lack of required scripting environments and the complexity of setting up software dependencies. Objective is the integration of advanced tools into existing applications and should be accomplished in a manner that allows both novice and experienced researchers to access these tools through the graphical user interface and interact with image inputs and processing results. One potential ad hoc solution could be the integration of these novel methodologies into interactive applications such as MITK or M²aia. A review of current efforts to successfully integrate new tools into existing applications reveals two potentially restrictive conditions. Firstly, executable scripts or command-line applications must be installed locally and all required dependencies must be configured correctly. Secondly, these tools must be integrated into the graphical user interface of the application. The integration of advanced tools into existing applications must be accomplished in a manner that allows both novice and experienced researchers to access these tools through the graphical user interface and interact with image inputs and processing results.

M²aia in its current state, focuses primarily on providing various interactive processing concepts for MSI datasets in imzML format. The scope of M²aia is therefore limited to the previously introduced MSI-related operations, such as the signal processing and 2D/3D MSI data interaction, and newly introduced concepts for interactive methods of multi-modal image fusion and 3D reconstruction. Nevertheless, there are many other interesting methods that would benefit from integration into the advanced interactive context of M²aia. For example, many specific tools for MSI data processing have been developed in R[72,73], while many deep learning approaches have been realized in Python[54,58,61,63,64,141]. Therefore, a programming language-independent integration of these methods into a platform such as M²aia would represent a considerable gain for the future development of the application.

This chapter presents a concept for integrating image analysis and processing methods into existing interactive frameworks to facilitate the adoption of cutting-edge technologies and improve the usability and functionality of applications in areas such as MSI. To achieve this, a solution is proposed that focuses on containerization. Containerization provides a robust framework for deploying and managing software without the hassle of dependency conflicts and environment setup issues. It allows for the consistent operation of applications across different computing environments, making it an ideal solution for integrating diverse (non-interactive) image analysis tools into interactive frameworks. These containerization technologies have matured significantly in the last few years, with Docker[i], as a prominent representative of this technology. Docker is openly accessible and is supported on Unix-based operating systems as well as on Windows. Details on Docker can be found in section 2.5 *Containerization with Docker*.

In addition to leveraging container technology, the proposed concepts include standardized interfaces for designing the user interface. These standardized interfaces ensure that new tools can be integrated with a consistent look and feel, which is crucial for maintaining a seamless user experience and control processing parameters

[i] https://www.docker.com/; accessed April 2024

through the graphical user interface. Furthermore, the concept defines the mechanisms for transferring parameters and data required for the execution of these tools. This involves specifying how data is passed to and from the containerized tools. Overall, the proposed integration strategy not only aims to simplify the incorporation of new image analysis methods into existing frameworks but also ensures that these powerful tools are more accessible to researchers in MSI and related areas.

3.4.2 Concept

This section outlines a concept for integrating third-party methods into interactive applications, regardless of the programming language used. The approach focuses on using file system operations to facilitate the transfer of runtime objects such as images from interactive applications to processing containers through intermediate data export operations. In the first part of the following description, the individual operational components necessary for this integration are described. The second part provides an overview of the operational workflow and introduces concepts of how these components interact to provide a seamless and efficient workflow for integrating third-party image processing methods into existing interactive frameworks.

Operational Components: the concept is built around four key components, each playing a critical role in the integration and execution of third-party image[i] processing methods:

1. Method-of-Interest: This is the specific image processing method that needs to be executed during the interactive processing session. It can be any algorithm or processing technique that enhances or analyzes the image data.

2. Process Docker Image: A custom Docker image that creates the environment for the process execution. It acts as the execution environment for the method-of-interest, ensuring that all operations are performed in an isolated and controlled setting. The process Docker image includes:

    - Dependencies: all the necessary dependencies required by the method-of-interest, eliminating the need for users to manually set up their systems.
    - Entrypoint Script: a script that facilitates process communication, managing the interaction between the container and the interactive application, and executes the image processing.

3. Interactive Application: the software used for image visualization and image processing, such as M²aia or MITK. This application is where users first interact with image data, performing tasks such as visualization, preliminary analysis, and data manipulation.

4. Custom View: A specifically designed interface within the interactive application that provides:

    - User Input: input elements for users to specify parameters and select runtime data, tailoring the processing to their specific needs.

[i] In the rest of this work, the term "image" is ambiguous. It can refer to an image of spatially distributed intensities, as in the context of image processing. Alternatively, it can refer to the definition of containerized runtime environments as used by Docker. Consequently, the term "Docker" is employed to describe any Docker-specific image.

- Process Trigger: initiate the execution of the process image, effectively linking user input with the method-of-interest.
- Data Handling: logic for preparing data for export to the Docker container and for managing the processed results once they are returned to the application.

Operational Workflow: the integration concept operates on a simple yet effective principle: if runtime data objects have an on-disk representation, entire folders containing these persistent data can be mapped as read-only into the container, allowing the container to access the data instantly without the need for copying. This setup is particularly beneficial for MSI datasets, where quick access to large-scale data is crucial.

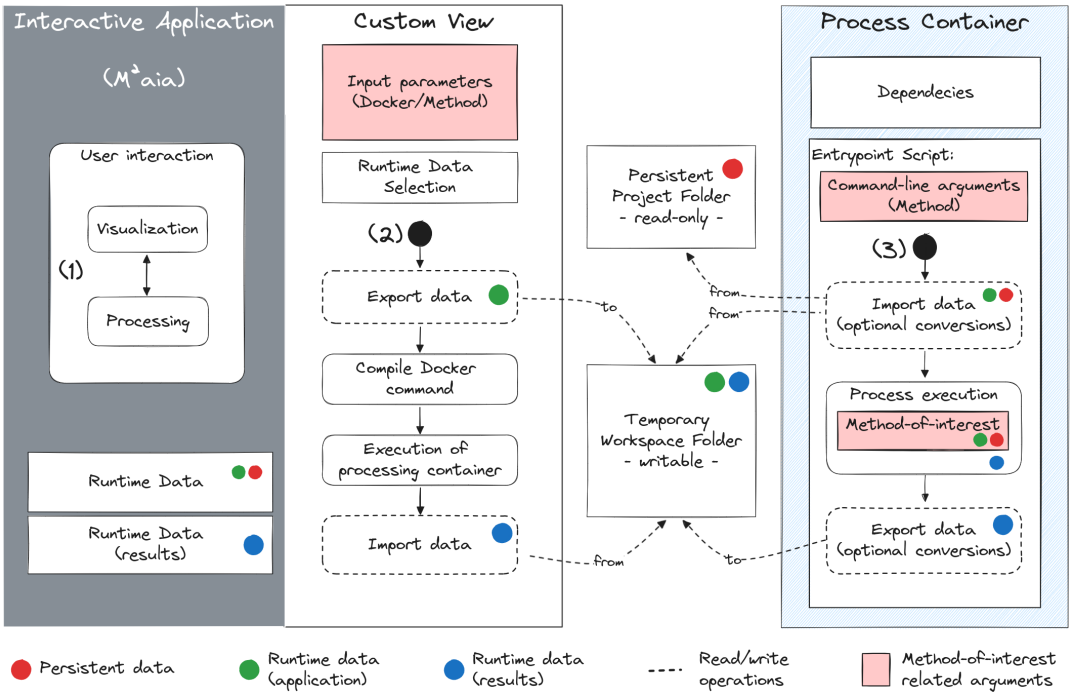

FIGURE 3.15. The figure outlines the concept of third-party method integration. Three main areas are shown: the interactive application on the left (gray), Docker-enabled processing on the right (blue), and the shared file system in the middle. Green dots indicate non-persistent runtime data generated by the interactive application. Red dots are runtime data objects with persistent representation. Blue dots are result data objects generated by the executed method-of-interest. Dashed lines/boxes are read/write operations on the shared file system. Red boxes indicate requirements/parameters related to the method-of-interest. The operational workflow is described in detail in subsection 3.4.2 *Concept*.

The interaction of the components that are essential for the integration of third-party methods in M²aia is illustrated in Figure 3.15 *Integration of Third-Party Image Processing Methods*. This illustration provides a comprehensive overview of how each component functions within the framework to facilitate Docker-based image processing methods using the interactive application, exemplarily demonstrated with M²aia:

- Interactive Application:

  - The process begins with (1) user interactions within M²aia, where images and associated files such as annotations or point sets can be visualized and manipulated interactively.
  - After successful completion of subsequent steps, the results become interactively available and can also serve as inputs for further executions.

- Custom View:
  - Once the decision has been made to initiate process execution, users can set parameters (red box) and select runtime data objects through the custom view's input UI elements. When a trigger event occurs, the process is initiated and the (2) custom view logic is executed. In a crucial initial phase, the data objects are exported and written to disk in a process-specific workspace folder (e.g., with a randomly generated ID), which can also be temporary (e.g., as a subfolder of the system's temporary data folder). It then compiles an executable Docker command. This command consists of command-line parameters used to run Docker and program arguments used to run the method-of-interest. The parameters defined in the Custom View are used. The Docker command is then executed.
  - In addition, the result data returned by the process container is also imported by the view logic and added to the interactive application as runtime objects.

- Process Container:
  - In the previous step, you requested execution of the Processing Container by running the compiled Docker command. This command requests a Docker image. If the Docker image does not exist, the requested Docker image is retrieved from a Docker image registry. This Docker image is then used to instantiate a process container. During instantiation, the compiled Docker command includes instructions to map individual folders from the host system to the container, such as the project and workspace folders. When the process container is started, the (3) entrypoint script is executed. It is used to facilitate communication based on command-line arguments (upper red box) and to execute the method of interest within the process container (lower red box).
  - Once the entry point script is started, the processing data is imported. In this step, the entrypoint script is used to prepare the data for the final execution of the method-of-interest. This script can access persistent data in read-only mappings (red dots) and data stored in the workspace folder (green dots). The data can optionally be converted to appropriate formats. Once the method of interest has been successfully executed, the entrypoint script is used to manage the final export of data (blue dots) to the workspace folder.
  - The results are written directly to the workspace folder.

3.4.3 Data Exchange Interface

To effectively integrate containerized applications within the graphical user interface of M²aia, a unified data access strategy is proposed. This strategy will facilitate the transfer of runtime data objects from M²aia to the containerized applications. Runtime data objects may have either a persistent representation (e.g. an imzML

image defined by the `.ibd` and `.imzML` files) or no persistent representation, thus only be accessible during its lifetime in M²aia.

In order to provide access to data within a containerized application, Docker's volume mapping is utilized. Volume mapping enables access to specific filesystem directories by mapping the content of one directory to one directory within the container. For standard data exchange an user-definable workspace folder is created, for each execution of a containerized application, on the local file system (e.g. in the system's temporary files directory) and is mapped into the container by default.

Furthermore the following key principles guide the development of this data interface:

- Direct Mapping of Unchanged Data: If a runtime data object in M²aia needs to be accessed by a containerized application without modification, the entire directory containing the original file is mapped into the container as read-only. This ensures that the original data can be utilized directly by the containerized application without any copy-overhead.
- Storing Processed Data: Runtime data objects (without persistent representation on the disk), e.g. results of processing steps generated within M²aia, are saved in the workspace directory. In order to write an object to disk, MITK's dynamic file writer utilities are used. The file is written to disk given a file format ending (like *.nrrd or *.png for images).
- Workspace for Output: Any results produced by a containerized application are saved by writing to the mapped workspace directory.
- Format Compatibility: It is essential that the data transferred between M²aia and the containerized applications be in compatible file formats. The responsibility for converting data to these compatible formats lies with the developers, who must decide whether the conversion will occur within the container or within M²aia.

These principles ensure a seamless integration of containerized applications with M²aia, allowing for efficient data exchange and processing continuity.

3.4.4 Process Integration

The process interface is based on passing command-line arguments to a containerized application using the `docker run` command[i]. The `docker run` command is separated into three parts. The first part contains Docker related attributes, like the instructions for volume mapping and graphic card usage flags. The second part is defining a Docker image. The third part contains the command-line arguments for the containerized application. A Docker image is defined for each containerized application, which provides a runtime environment tailored to an executable script (e.g. a Python or R script). This application image contains all required dependencies for execution.

These containers are only used once and removed after execution. The application images are to be designed in such a way, that the executable script is executed directly after the container startup was successfully. This can be realized by using the execution command of the script as an entrypoint[ii]. This enables the parametrization and

[i] https://docs.docker.com/engine/reference/run/; accessed April 2024
[ii] https://docs.docker.com/reference/dockerfile/#entrypoint; accessed April 2024

execution of a script via the command-line by adding file paths and other parameters directly to the `docker run` command as additional arguments.

In order to run an application container a helper class is conceptualized, supporting the afore mentioned principle for data exchange and the execution of an application image in a developer-friendly way. The main task of the helper object is to provide an intuitive interfaces to prepare and execute an application image specific `docker run` command. The usage of this helper object is demonstrated exemplary in Listing 3.4 *Exemplary usage of the Docker helper class* and demonstrates how the docker image parameters are set. The executed `docker run` command is shown for this example in Listing 3.5 *Exemplary docker run command generated by the docker helper class shown in Listing 3.4*.

Finally, the helper class runs the application image and tracks the status of the execution. Once the execution has ended, a list of `mitk::BaseData` objects is created, by loading the defined output files from disk. These objects can be added to the DataStorage view to enable interactive access within M²aia.

Although the focus is on M²aia, the concepts are also transferable to other applications. The concepts presented here (and their implementation) can be used as a blueprint for corresponding developments. All features for the integration of third-party applications within a MITK-based interactive application such as M²aia are realized as an external project using the MITK project template. The respective MITK extension will be referred to as `mitk-docker`[i].

```
// Docker image used for image processing
std::string image = "ghcr.io/m2aia/anyimage:latest"

// The docker helper class prepares and executes the docker run command
mitk::DockerHelper helper(image);

helper.EnableAutoRemoveContainer(true); // delete the container after it was executed
helper.EnableGPUs(false); // enable usage of graphic cards within the container

// container input
helper.AddAutoSaveData(imzMLDataNode->GetData(), "--imzml", "file_msi",".imzML");
helper.AddAutoSaveData(centroidNode->GetData(), "--centroids", "file_centroids",".centroids");

// container output
helper.AddAutoLoadOutput("--output_0", "output_0.nrrd");
helper.AddAutoLoadOutput("--output_1", "output_1.nrrd");

helper.AddApplicationArgument("--param_0", m_Controls.param_0->text().toStdString());
helper.AddApplicationArgument("--param_1", m_Controls.param_1->text().toStdString());
helper.AddApplicationArgument("--param_2", m_Controls.param_2->text().toStdString());
helper.AddApplicationArgument("--param_3", m_Controls.param_3->text().toStdString());

// start processing
const auto results = helper.GetResults();

for(auto result : results){ // do something with the results ...}
```

LISTING 3.4. Exemplary usage of the Docker helper class. This C++ code snippet demonstrates the use of the mitk::DockerHelper class to prepare and execute a Docker container. It configures the container to automatically remove itself after execution and disables GPU usage. The script sets up input data files with specific arguments and specifies the output files to be automatically loaded after processing. Application-specific parameters are passed to the container, and finally, the results of the container execution are retrieved.

[i] https://github.com/m2aia/mitk-docker

```
docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]

docker run --rm \
    --volume /local/absolute/path/to/workspace:/container/workspace \
    ghcr.io/m2aia/anyimage:latest \
    --param_0 value_0 \
    --param_1 value_1 \
    --param_2 value_2 \
    --imzml /container/workspace/file_msi.imzML \
    --centroids /container/workspace/file_centroids.centroids \
    --output_0 /container/workspace/results/output_0.nrrd \
    --output_1 /container/workspace/results/output_1.nrrd
```

LISTING 3.5. Exemplary docker run command generated by the docker helper class shown in Listing 3.4.

3.5 Concepts for Deep Learning on MSI data

In this section the imzML image access capabilities are extended for deep learning applications. In section 3.5.1 an problem overview is introduced along with general concept details in 3.5.2. Further specified are concepts for data access strategies in section 3.5.3, key features of the batch generators in 3.5.4 and the complementary details for shared library access in section 3.5.5.

Parts of this section have been published in:

- *Cordes et al. pyM2aia: Python Interface for Mass Spectrometry Imaging with Focus on Deep Learning. Bioinformatics (2024)*[118]

3.5.1 Overview

The slow development of deep learning in the processing of MSI data can be attributed to various factors. These include the lack of standardized benchmark datasets[20], inconsistent data quality due to batch effects[36], a wide range of signal processing approaches for MSI data[18,54], challenges arising from the curse of dimensionality, and the untraceability of deep learning models that lack explainability and interpretability (often referred to as "black box" models). Furthermore, the state-of-the-art in deep learning often relies on widely used deep learning frameworks in Python, such as TensorFlow[i] or pyTorch[ii]. However, Python packages for the specific provision of spectral data in the context of deep learning model development for MSI do not exist. There are two packages for loading imzML data in Python[142]. These are pyImzML[iii] and imzml-rs[iv]. pyImzML is an optimized package for raw imzML[53] data import and export, but is lacking signal processing capabilities. It is used by the well known METASPACE[84] online platform for molecular annotations.

Previous Python-based open source implementations of deep learning applications in the field of MSI have explored various approaches to address data access. Abdelmoula *et al.* (2021)[63] realizes an unsupervised method for the identification of peaks which is based on an upstream conversion step of the MSI data into the hdf5[v] format[63,64]. Hu *et al.* (2022)[58] and Ovchinnikova *et al.* (2020)[57] use databases of already exported 2D ion-images for training of deep learning models for ion-image

[i] https://www.tensorflow.org/; accessed April 2024
[ii] https://pytorch.org/; accessed April 2024
[iii] https://github.com/alexandrovteam/pyimzML; accessed April 2024
[iv] no documentation found; April 2024
[v] https://www.hdfgroup.org/; accessed April 2024

co-localization tasks[57,58]. Li *et al.* (2023)[141] also uses a preprocessed dataset in combination with OpenCV[i] to load data for the training of a deep learning model for accelerating 3D image reconstructions. All of the methods avoid the direct use of imzML data. In addition, the reproducibility of the pre-processing of MSI data is often impaired by a lack of documentation or the use of unpublished processing techniques. It would therefore be advantageous to process raw data directly.

The hypothesis is that employing fast and memory-efficient processing techniques tailored to the imzML file format could significantly enhance the training and development of deep learning models by supporting optimized and convenient data access interfaces. The imzML standard not only allows for the storage of raw data but also facilitates the documentation of preprocessing steps directly within the file format. As a well-established community standard, imzML offers extensive capabilities for FAIR (Findability, Accessibility, Interoperability, and Reusability) data exchange, as highlighted in the literature[76,143].

3.5.2 Concept

A unified code base that spans interactive exploration and scripting applications can ensure consistent data visualization and processing across both domains. By enabling seamless transitions of data between interactive applications and a Python-based scripting environment, tasks such as the interactive creation of spatial annotations and interactive visualization of image inputs, intermediate results, and outputs are streamlined. This integration could be achieved by encapsulating M²aia's optimized imzML data import strategies within a Python framework. Consequently, image data examined in the interactive M²aia application should be accessible in Python and vice versa, ensuring a fluid interchange of data between these platforms. This facilitates a more integrated workflow, allowing for both detailed analysis within M²aia and flexible scripting and automation in Python.

To optimize deep learning implementations for MSI, the integration of M²aia's imzML import directly into a Python based environment could offer significant benefits. The goals of this integration are: (i) Enabling efficient access to individual spectra and ion-images so that deep learning models can process the data with increased speed and flexibility; (ii) utilising M²aia's advanced signal processing capabilities, which are closely related to lazy loading techniques; (iii) streamlining metadata queries directly from the imzML format to ensure that relevant data attributes can be retrieved quickly and easily; and given that MSI datasets are represented as a three-dimensional data cube—with two spatial dimensions and one spectral dimension, which can expand extensively due to substantial spectral bandwidth—it is crucial to (iv) define and develop tools that realize data access strategies that support targeted queries. Such data access strategies are essential in deep learning methods to reduce the data in the input layers of the networks. Focusing on reducing the field-of-view can significantly enhance data management, enabling more efficient processing and analysis for deep learning applications. This approach narrows down the amount of data that needs to be processed at any one time, potentially speeding up computations and reducing the resource load. However, this method also introduces a trade-off: while it decreases the data load, it simultaneously requires a compromise between the completeness of the data and its manageability. By limiting the field-of-view, important features or patterns could be omitted, which might impact the accuracy or the effectiveness of the deep learning models. Thus, finding the right balance between reducing the field-of-view and maintaining the integrity and completeness

[i] https://pypi.org/project/opencv-python/; accessed April 2024

of the data is crucial for optimizing both performance and outcomes in deep learning tasks.

By utilizing the memory-efficient and fast imzML import functions of M²aia together with the computationally intensive signal processing functions, Python-based access to imzML data can be enabled. This is achieved by integrating the shared library functions of M²aia directly into a Python context. For this purpose, a wrapper class is designed that enables functions from the M²aia shared library to be called.

This wrapper object is designed to provide convenient access to essential elements in imzML datasets (points i-iii), such as metadata, spectra and ion-images, while providing advanced signal processing techniques. By integrating these functionalities into a single wrapper class, users can effortlessly interact with MSI datasets. This optimized approach not only simplifies the handling of large datasets in Python, but also improves the efficiency of data analysis and processing, making it a powerful tool for researchers and developers.

Additionally, classes that enable deep-learning-oriented data access (point iv) are crucial. These classes should support targeted queries to reduce the field-of-view and also be capable of performing data transformations, augmentations, and batch generation. In order to work out the classes in more detail, the concepts for data access are explained in the following section. The class concepts and the resulting Python package are referred to as pyM²aia[118].

3.5.3 Data Access Strategies for MSI Data

As discussed in the previous section, the complexity of MSI poses a challenge for its use in training deep learning models. Looking at the current state-of-the-art in MSI data handling, different access strategies have been identified. These include strategies that only access spectra, those that only retrieve ion images, and hybrid approaches that allow combined access to spatial-spectral sub-cubes. These strategies are important to reduce the amount of data for training, as complete datasets, as mentioned, cannot yet be used directly as input for deep learning models due to their size. These access strategies for MSI have not yet been formally documented.It is hypothesized that categorization of methods according to these access strategies benefits the communication about MSI deep learning approaches. The following basic data access strategies are proposed:

Spectral Strategy The spectral strategy uses spectral information only and discards the spatial relationships between spectra. Different models exist utilizing this strategy, including spectrum-wise peak picking and classification by Abdelmoula *et al.* (2021)[63] or dimensionality reduction methods by Thomas *et al.* (2016)[144]. The spectral strategy is illustrated in Figure 3.16 *Spectral Strategy*. This strategy actually pursues the use of entire spectra, but it can also be useful to consider only predefined positions on the m/z axis, e.g. to reduce the amount of data or to find specific correlations within a selection of m/z positions.

Spatial Strategy The spatial strategy addresses the spatial properties of a molecular distribution, but intra-spectral relationships are not taken into account. Different models exist including the work of Hu *et al.* (2022)[58] and Ovchinnikova *et al.* (2020)[57] for ion-image co-localization tasks[57,58]. For these tasks, usually only a small subset of the available m/z values and the corresponding intensities are relevant, which are ultimately required for ion-image generation (see subsection 3.2.3 *Ion Image*

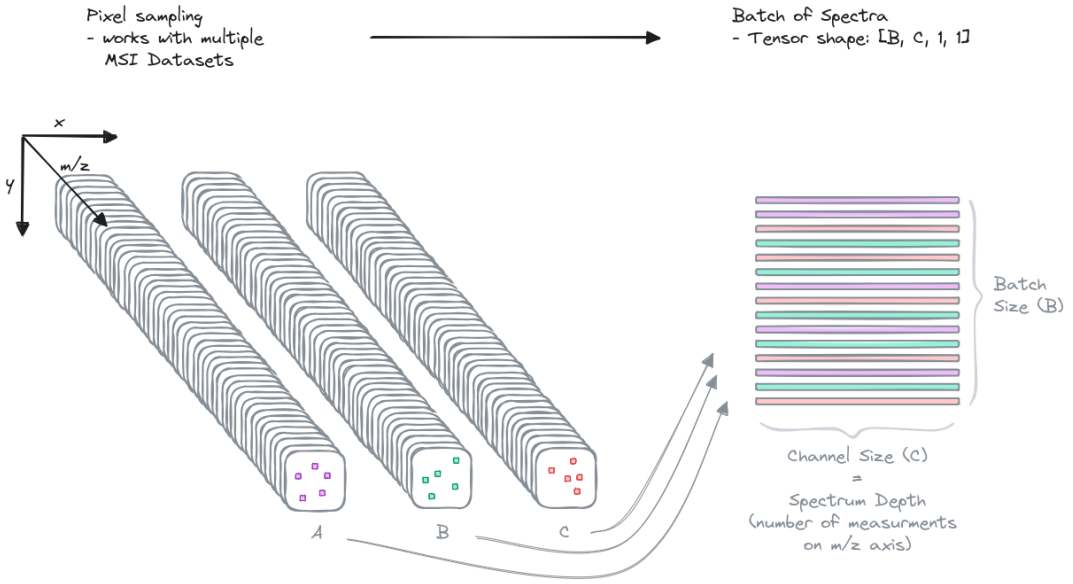

FIGURE 3.16. Illustration of the spectral strategy within the context of batch generation in deep learning applications. On the left, three different MSI datasets (A,B,C) are randomly sampled in the spatial domain (colored squares). Sampled spectra are stacked to provide a batch of elements with size B and channel size C.

Generation). A corresponding limitation could be realized by providing m/z values of individual peaks that exceed a certain noise-to-signal ratio or that are relevant in the context of an investigation. The spatial strategy is illustrated in Figure 3.17 *Spatial Strategy*. In order to apply the spatial strategy to multiple MSI datasets simultaneously, it is necessary to consider how to standardize the ion-images, e.g. by cropping them.

Spatio-Spectral Strategy The spatio-spectral strategy uses spatial and spectral information simultaneously, which are computationally highly demanding and still rare. The spatial strategy is illustrated in Figure 3.18 *Spatio-spectral Strategy*. As with the spatial strategy, it often makes sense not to use the entire spectral depth and to restrict it by specifying a list of m/z values. However, this can vary in this strategy depending on the question. To select spatially adjacent spectra, a neighborhood can be defined around a central pixel. Finally, this leads to spatial-spectral samples that may not have the full spectral depth and include close neighbors.

3.5.4 Deep Learning Support by Batch Generation

To generate batches of samples for training and inference of deep learning models, pyM²aia introduces appropriate data structures. A unified batch generation process is based on the so-called `SpectrumDataset` and `IonImageDataset` for spectral and ion images, respectively. The general purpose of a dataset is to generate samples that can be compiled into batches for training/inference. These datasets are designed to handle multiple imzML images simultaneously. The main features of the SpectrumDataset and IonImageDataset are listed below.

SpectrumDataset The `SpectrumDataset` aims to provide convenient access to spectral or spatio-spectral samples derived from single or multiple imzML datasets. The concept is illustrated in Figure 3.19 *Concept for batch generation using the SpectrumDataset*. The key features of the SpectrumDataset are:

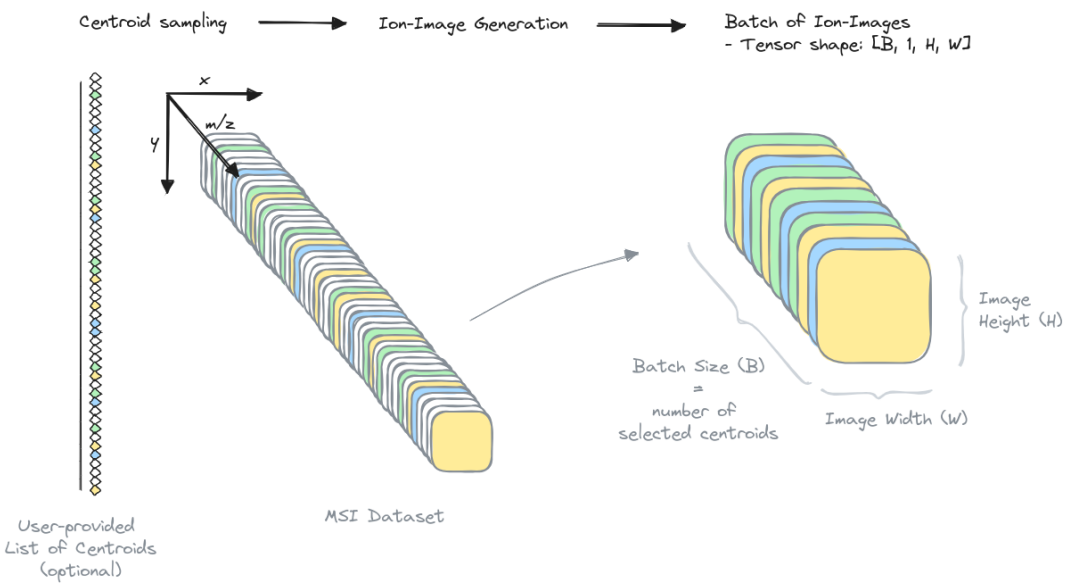

FIGURE 3.17. Illustration of the spatial strategy within the context of batch generation in deep learning applications. On the left, a MSI datasets is illustrated. Given a user-provided list of centroids (m/z values), ion-images are randomly sampled in the spectral domain (colored squares). Sampling is restricted to the list of centroids. Sampled ion-images are stacked to provide a batch of elements with size B, image width W, and image height H.

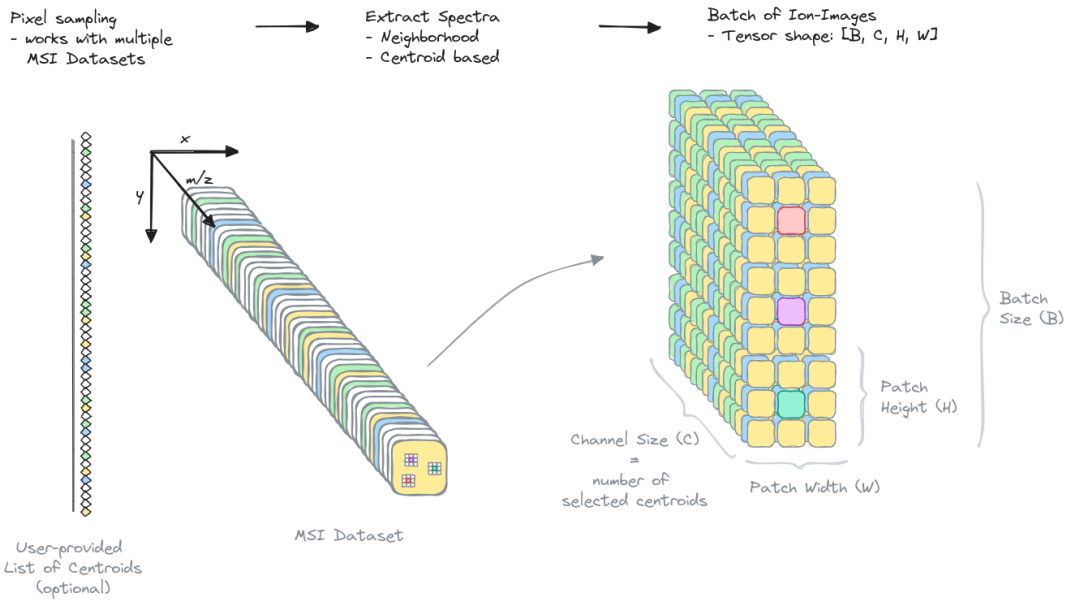

FIGURE 3.18. Illustration of the spatio-spatial strategy within the context of batch generation in deep learning applications. On the left, a MSI datasets is illustrated. Subregions are randomly sampled in the spatial domain (red, purple, turquoise squares) by simultaneously sampling the spectral domain by a given user-provided list of centroids (m/z values; yellow, green, blue squares). Sampled subregions are stacked to provide a batch of elements with size B, channel size C, region width W, and region height H.

- Spectral Strategy: The dataset returns multiple spectra and associated label per query.
- Spatio-Spectral Strategy: The dataset returns a single spectrum, associated label and adjacent neighboring spectra per query.
- Consistent Spectral Depth: In order to handle multiple imzML images a consistent spectral depth across all images is guaranteed.
- Support for Labels: A label mask can be included to assign labels to each spectral sample, facilitating the training of supervised deep learning models.
- Query Neighboring Spectra: For the spatio-spectral approach, a specific shape element is necessary to define a neighborhood around a pixel position (patches).
- Centroid List Provisioning: Users can generate specific sub-spectra by providing a list of m/z values. This tailored approach ensures that the generated spectra are relevant to the user's specific research needs.
- Spectral Buffering: For both, the spectral strategy and spatio-spectral strategy, an in-memory buffering of sampled spectra can be enabled. This is realized using a key-value store with the spectrum indices as keys and the intensity values of the spectra as values. If a spectrum are is queried again, e.g. in a subsequent training epoch, spectra can be queried directly from the key-value store.

IonImageDataset The `IonImageDataset` is specifically designed to facilitate the retrieval and manipulation of ion-image samples derived from single or multiple imzML datasets. The concept is illustrated in Figure 3.20 *Concept for batch generation using the IonImageDataset*. The following key features are summarized:

- Spatial Strategy: The dataset returns multiple ion-images per query.
- Flexible Data Handling: Continuous/processed-centroid and continuous-profile imzML images with different spectral depths and non-uniform image dimensions are supported.
- Centroid List Provisioning: Users can generate specific ion-images by providing a list of m/z values. This tailored approach ensures that the generated images are relevant to the user's specific research needs.
- Fallback for Non-existent Data: In instances where the requested m/z values fall outside the spectral range of the imzML dataset, the dataset will generate a blank image. This feature ensures that the dataset's integrity is maintained even when specific data points are unavailable.
- Image Transformation: To standardize the diverse dimensions of ion-images for computational analysis, the dataset allows for sample transformations. These transformations may include central cropping or resizing, adapting the images to a uniform size suitable for further analysis[i].

[i] https://pytorch.org/vision/stable/transforms; accessed April 2024

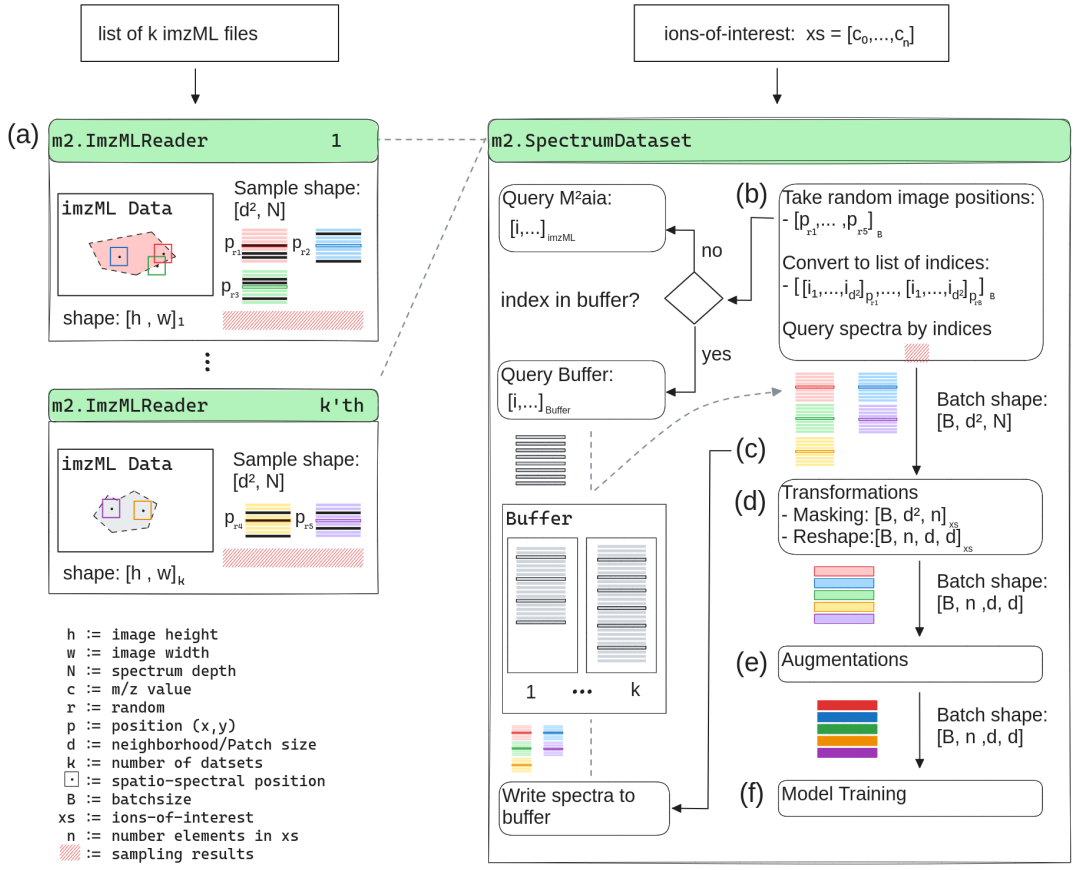

FIGURE 3.19. Schematic representation of spatial-spectral data access. In the initial phase, (a) up to k reader objects (`m2.ImzMLReader`) are created. Signal processing can be configured individually for each of the reader objects. The images are assigned to the dataset (`SpectrumDataset`). The stack generation starts with (b) the random selection of B valid pixel positions $P = [p_{r1}, ..., p_{rB}]$, across all images, with B as the batch size. To follow the spatial-spectral strategy, patches of size $[d, d]$ are generated around the x-y positions p_r. The index image of M²aia is used (see section 3.2 *Concepts for MSI Data Processing*) to select all spectra within the patch with the centre position defined by p_r. For each patch, $[i_1, ...i_{d^2}]$ indices are retrieved. Each spectrum index is checked to see whether it has already been retrieved in a previous iteration. An in-memory buffer can be used to keep track of individual queries in order to generate final query lists of indices: a list $[i, ...]_{imzML}$ for querying the MSI records and a list $[i, ...]_{buffer}$ for querying from the buffer. All spectra are now either generated by M²aia or loaded from the buffer. Once all spectra have been retrieved, the unprocessed and unbuffered spectra (c) are stored in the buffer. Next, the spectra (d) are transformed, such as by masking or reshaping. Masking is optional and can be used with a user-defined list xs of m/z values. The aim is to reduce the spectral bandwidth and data load. If $d > 1$, the list of spectra is reshaped to obtain patches of size $[d, d]_{p_r}$. Extensions (e) can be applied to manipulate the stacks directly before (f) training the model parameters. In the case of $d = 1$, the procedure corresponds to the spectral strategy.

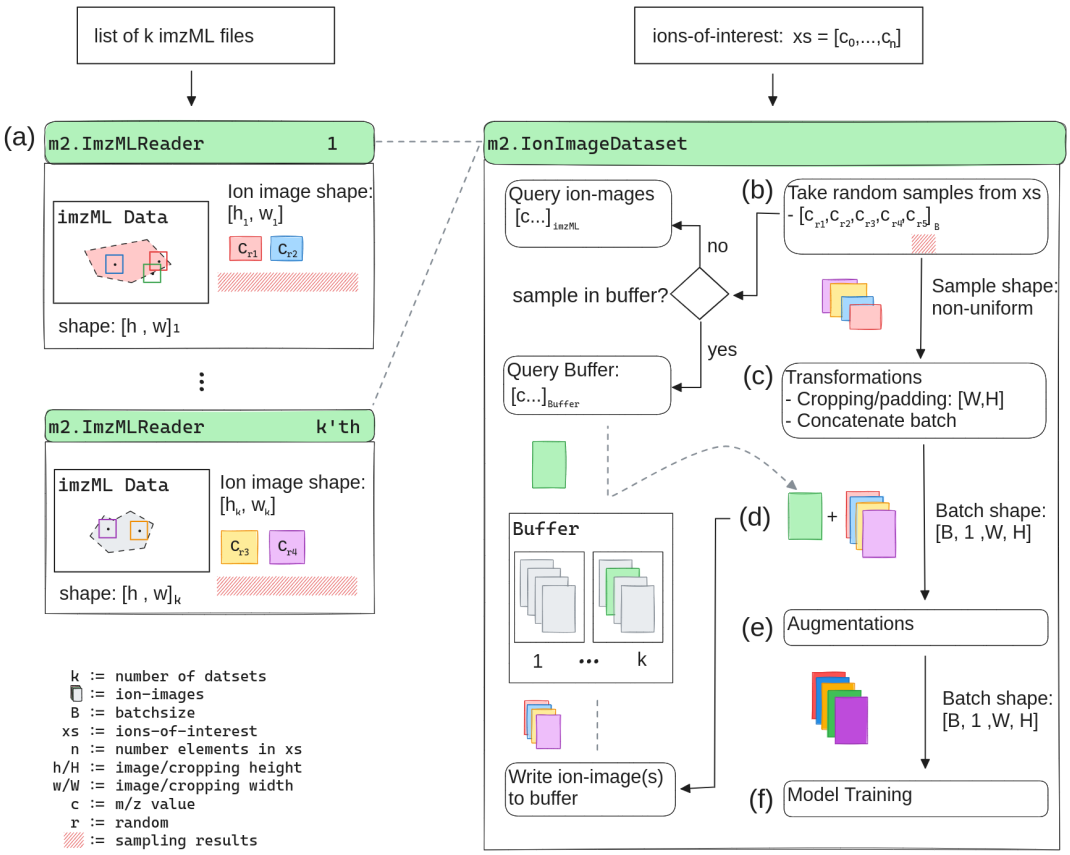

FIGURE 3.20. Schematic illustration of spatial data access. In the initial phase, (a) up to k reader objects (`m2.ImzMLReader`) are created. Signal processing can be configured individually for each of the reader objects. The images are assigned to the dataset (`IonImageDataset`). The stack generation starts with (b) the random selection of B m/z values $[c_{r1}, ..., c_{rB}]_{xs}$, from the user provided list xs, with B as the batch size. To follow the spatial strategy, ion-images are generated for each c_r. The ion-image generation capabilities of M²aia are used (see section 3.2 *Concepts for MSI Data Processing*) to generate ion-images. An in-memory buffer can be used to keep track of generated ion-images: a list $[i, ...]_{imzML}$ for querying the MSI records and a list $[i, ...]_{buffer}$ for querying from the buffer. In order to handle the different image sizes of ion-images generated by the different reader objects, (c) ion-images are transformed using methods such as central cropping. The results of the transformation is a uniform batch of ion-images. Additionally, the ion-images from the buffer have now the same shape and can be concatenated. Unbuffered ion-images are (d) written to the buffer. Image augmentations (e) can be applied to manipulate the stacks directly before (f) training the model parameters.

- Augmentations: The application of augmentations is supported. These augmentations can enhance the dataset's utility in deep learning models by introducing variability that helps in generalizing the model's performance across different data scenarios.

- Ion Image Buffering: The ion-image buffering is realized as on-disk buffering. Ion-images are drawn from the imzML images and the image transformations are applied, e.g. in order to generate ion-images with similar shape. These transformed ion-images are stored on disk, e.g. as numpy binary files. If an ion-image is queried again, e.g. in a subsequent training epoch, the ion-image can be loaded from disk. This way, the, possibly data and computation intensive, ion-image generation process (see subsection 3.2.3 *Ion Image Generation*) and the image transformations are skipped.

Both the `IonImageDataSet` and the `SpectrumDataset` classes are designed to handle multiple imzML images, facilitating a comprehensive approach to data management in deep learning MSI applications. Access to M²aia-based `m2::SpectrumImages` — detailed in section 3.2 *Concepts for MSI Data Processing* and illustrated in Figure 3.5 *Class diagram of handling hyperspectral images in M²aia*—is integrated into the `m2.ImzMLReader`. This class serves as the interface to these datasets, covering aspects (i-iii; defined in Figure 3.21 *Access to shared library*) which include the efficient extraction and handling of metadata, spectra, and ion-images from imzML datasets. This structured approach ensures that all data types necessary for comprehensive analysis and machine learning applications are readily accessible and efficiently managed within the Python environment. An overview of the concept of using pyM²aia for the training of deep learning models is illustrated in Figure 3.21 *Access to shared library*.

3.5.5 Access to imzML Datasets

The Datasets described in subsection 3.5.4 *Deep Learning Support by Batch Generation* utilize the `m2.ImzMLReader` component of pyM²aia, a specialized reader designed to handle imzML files efficiently. This reader integrates the shared functions from M²aia's C++ libraries, enabling data processing capabilities directly in the Python environment. The `m2.ImzMLReader` offers a comprehensive set of features, allowing users to access the imzML spectral formats, such as continuous and processed profiles or centroids. It supports all of M²aia's imzML import and data access concepts as the signal processing options, accesses to overview spectra and helper images, and the ion-image generation as well as querying single spectra using indices. Additionally, the metadata of imzML datasets can be accessed rapidly ensuring applicability to data management processes. The integration with Python is facilitated by ctypes[i], a library used for calling functions from DLLs or shared libraries. Setting up the `M2AIA_PATH` environment variable is critical as it defines the search path for these libraries, allowing users to utilize either the pre-packaged binaries or customized versions of the M²aia libraries. Summarized, the key features of `m2.ImzMLReader` are:

- Supports all imzML-defined spectrum formats.

- Provides extensive signal processing capabilities.

- Efficient access to overview and individual spectra.

- Rapid ion-image generation capabilities.

[i] https://docs.python.org/3/library/ctypes.html; accessed April 2024

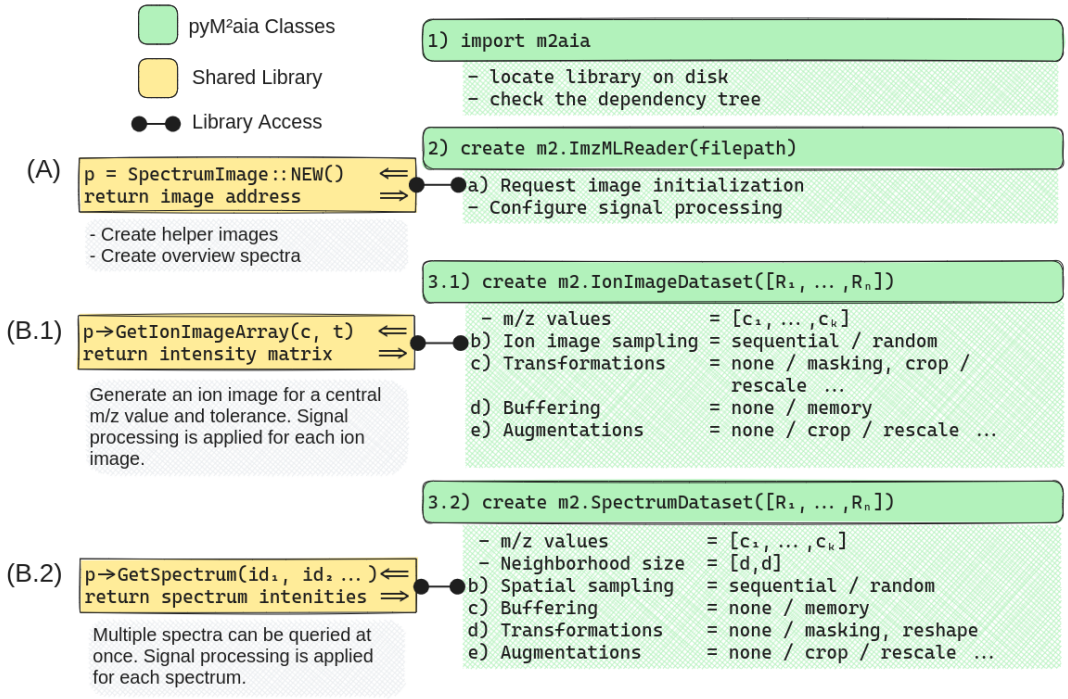

FIGURE 3.21. This illustration highlights the shared-library access during batch generation for the spatio-spectral (see Figure 3.19 *Concept for batch generation using the SpectrumDataset*) and spatial (see Figure 3.20 *Concept for batch generation using the IonImageDataset*) access strategies. On the left side, the functions of the shared library are illustrated (yellow boxes). These are used by pyM²aia objects to query data from imzML reader objects within the scope of M²aia. In order to execute appropriate queries, the (1) initialization of Python package (`import m2aia`) is used to check for a valid shared library object (using system variable `M2AIA_PATH`) and further evaluates all required dependencies. ImzML reader objects (`m2.ImzMLReader`) are (2) initialized with a given file path variable. These readers can be configured in order to execute signal processing steps during the access of the imzML data. The created (A) image address is returned and stored as access point within the reader object. In (3.1) an ion-image dataset (`m2.IonImageDataset`) and in (3.2) an spectrum-dataset (`m2.SpectrumDataset`) are created and initialized with multiple reader objects R_i, with $i = 1, ..., n$. During the sampling of the reader objects (B.1) the shared library is used to execute the ion-image generation in order to provide access to the intensity matrix. In (B.2) multiple spectra are queried simultaneously for data sampling using the spectral strategy. The black lines with circles indicate at which step of the sampling data are queried from M²aia.

- Fast access to imzML metadata facilitating data management processes.

Chapter 4

Concepts for Interactive Remote Working

In this chapter, the concepts for an interactive Remote Working Environment (RWE) are introduced for processing of MSI datasets. An overview of the problem and research objectives are provided in section 4.1, the targeted user-group and requirements are introduced in section 4.2. The platform architecture is introduced in section 4.3 and the packaging of interactive applications for the biomedical image processing, based on the containerization technology Docker, in section 4.4. The use concept of these applications is finally described in section 4.5.

4.1 Overview

In the rapidly evolving field of biomedical research, the ability to interactively access images is crucial for effective analysis, methodological innovations and collaborative endeavors. As imaging techniques continue to advance, generating ever more detailed datasets, the amount of data to be stored and processed will continue to increase. As a result, data and computationally intensive processes are therefore increasingly being shifted to remote resources. Large, multidimensional and multimodal biomedical images, as generated by Mass Spectrometry Imaging (MSI) (see section 2.1 *Mass Spectrometry Imaging*), pose major challenges to a fast, comprehensive and interactive access of remote images. Processes as image analysis, development of new methods, and interdisciplinary collaboration of domain experts can be hampered if data-intensive transfers to local systems are required, e.g. for processing of images with interactive desktop applications the domain experts are familiar with. Current efforts to utilize remote resources focus on providing IDEs for remote development and applications for execution of reproducible image analysis workflows (see section 2.3 *Bioinformatics*). However, the alternating steps of a development process, which include code generation, code execution, and full interactive validation of images and image-related inputs and results, are difficult to perform completely on remote resources. One reason for this is that interactive applications in the field of biomedical image processing are often not designed for remote working. Experience, interviews with researchers and analysis of existing workflows show that a comprehensive interactive remote access, particularly to complex data sets such as those generated by MSI, would be beneficial. Therefore, in this work the provision of suitable solutions that can help us to carry out the entire development and analysis processes remotely, focusing on MSI-related applications, is pursued.

The following three research questions are addressed: (1) Can interactive applications, like M²aia, be hosted on remote resources to process image data for which there are no interactive remote solutions yet, e.g. MSI data? (2) Can the interactive

image exploration and thereby the development process of new image-based processing methods be supported or/and completely shifted to remote resources? (3) Can interactive access to image processing applications be realized in a simple, efficient and user-friendly way?

In this chapter, the concept of a flexible and extensible remote working environment is introduced, where method development, data processing, collaboration, and especially interactive processing tasks, are performed by users directly on a remote machine (server or in the cloud). This developer-friendly environment should result in the complete avoidance of transferring entire data sets to the user's local system.

Definition: An interactive Remote Working Environment (RWE) is a software eco-system that enables interactive and collaborative development tasks on remote resources. Remote working includes the following tasks:

- Remote Execution: permits the execution of source code, processing applications and the management of interactive applications. They can run computationally intensive tasks, avoiding the limitations of local machines and improving overall efficiency. Execution privileges are to be regulated by server-execution-policies.
- Remote Interactivity: provides users and collaborators to access interactive applications, allowing for rapid data exploration, image annotation, and easier sharing of data related processing tasks, insights and results. The interactive applications can be hosted in read-only mode to act as a viewer application or in read-and-write mode for data related processing.
- Remote Development: aims to develop new methods directly on remote located resources. This may require utilization of remote interactivity and remote execution tasks. The entire source code, application dependencies and required (image) datasets are provided on the server side. During development, it may be necessary to make corresponding data interactively accessible, for example to view new data immediately after it has been recorded, to verify intermediate results of remote executions, or to make individual data objects or collections available to collaborators.

4.2 User Groups

The interactive RWE is primarily aimed at two user groups: Data Analysts and Data Users. A Data User is a person who focuses on specific interactive tasks, including the exploration of existing datasets or the generation of new data such as image annotations. A Data Analyst, on the other hand, is a specialized Data User who follows a research question, that often additionally requires the creation or adaption of source code for new algorithms and applications. In addition, Data Analysts are able to transfer data to and from remote storage devices. They may provide interactive access to images or image-related data, both for themselves and for collaborators, through interactive applications. In Figure 4.1 *Interactive Remote Working Environment - User Groups*, user actions for remote developing, remote execution and remote interactivity are illustrated.

The basic idea behind the interactive RWE is to provide interactive access to imaging and related data during all stages of the method development process. A typical user scenario that should be targeted is remote collaborative method development with biomedical imaging datasets as illustrated in Figure 4.2 *Interactive Remote Working Environment - User Scenario*.

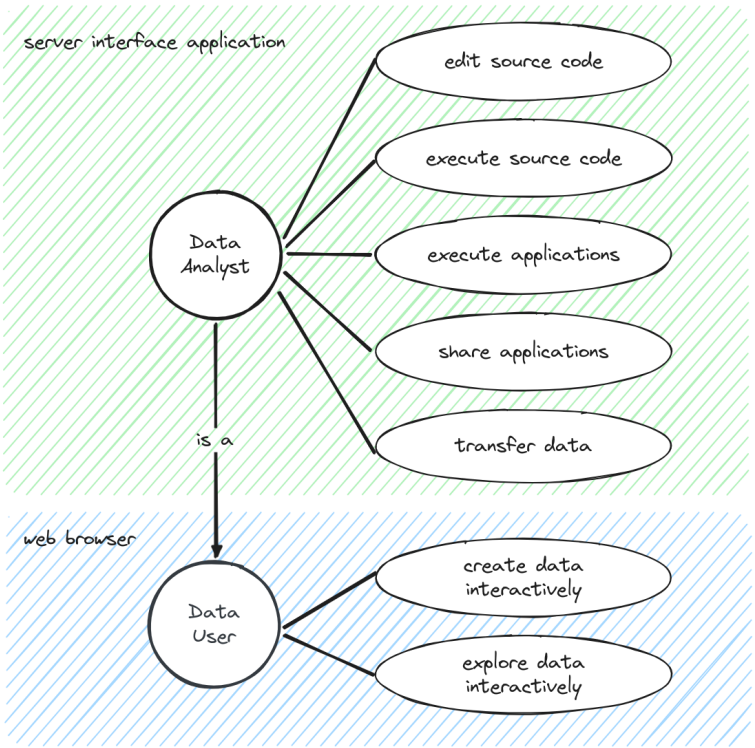

FIGURE 4.1. Interactive Remote Working Environment - User Groups: actions are executed within different interactive interfaces. The server interface application is the primary interface for remote developing and remote execution is also used to manage interactive applications (green). Interactive applications are remote executed applications that can be accessed using a standard web browser (blue).

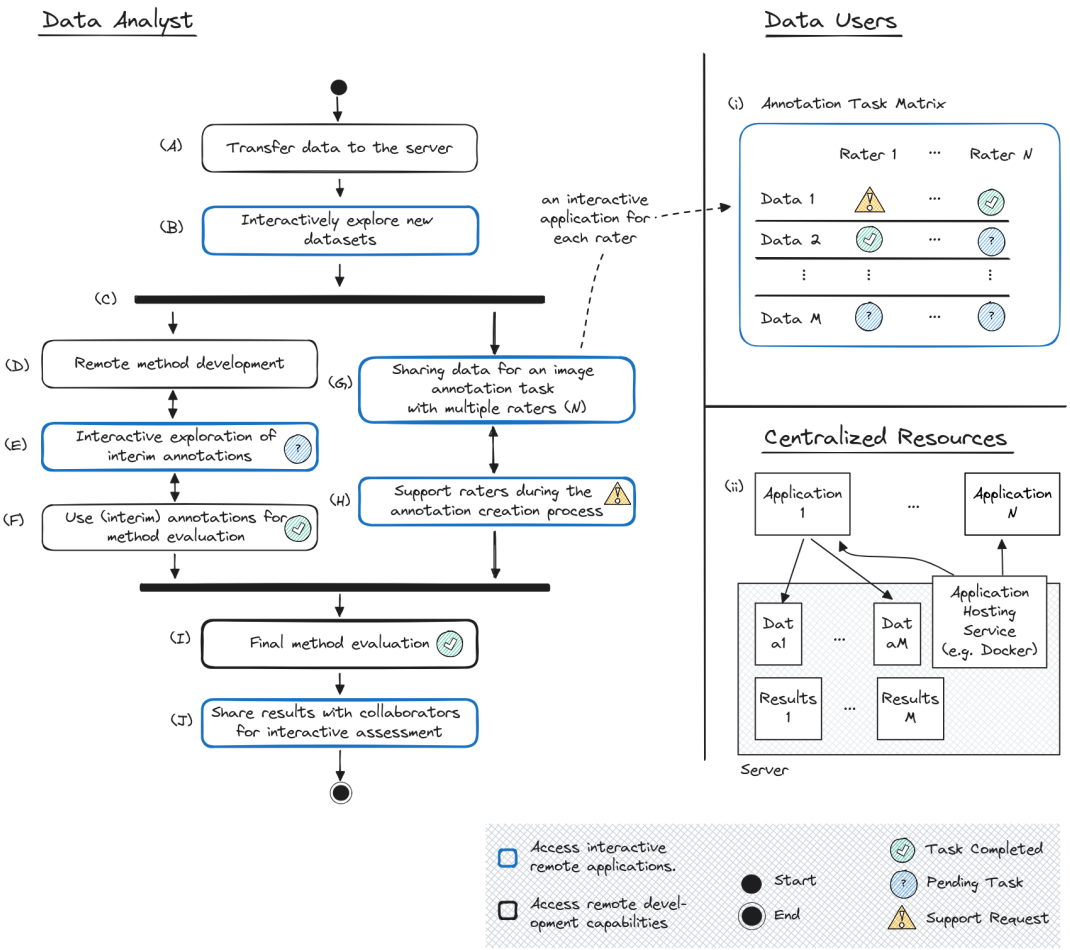

FIGURE 4.2. Interactive Remote Working Environment - User Scenario: This example user scenario illustrates actions proposed for a development process of new image-based methods using collaboratively generated data. A task sequence for the Data Analyst is shown on the left. The Data Users and a task matrix for completed, pending, and support tasks are shown on the top right. On the lower right are the centralized resources, showing hosted applications, image datasets, and the hosting service for the applications on a server. **Data Analyst:** once the data transfer to the server is complete (A), an interactive application can be hosted on the server to explore the image datasets (B). The process can now be split (C) into two parallel paths: following the left path, the Data Analyst can start remote development processes (D-F), including alternating steps of development, interactive exploration, and method evaluation. During development, the Data Analyst can access intermediate results of annotations generated by Data Users (raters 1 to N). Following the right path, the Data Analyst can (G) host interactive applications (Application 1 through N) for each Data User using a hosting service (ii) to provide the Data Users with access to all image datasets (Data 1 through M). The Data Analyst can simultaneously access the Data Users' applications to (H) assist in the image annotation process, e.g. when questions arise. Once the annotation process is completed for all image datasets, the final method evaluation can be realized (I). Finally, the results of the developed method (J) can be made available to other Data Users for demonstration purposes. **Data Users:** receive a web addresses to remotely hosted applications to start annotation processes. Data Users can request assistance from the Data Analyst. The annotation task matrix (i) indicates the status of the collaborative process.

Requirements

Based on the exemplary user scenario (see Figure 4.2 *Interactive Remote Working Environment - User Scenario*) and analysis of existing workflows and with regard to the research environment this work was created, the proposed interactive RWE is conceptualized with the following requirements in mind:

Integrability The interactive RWE should be designed as an extension of the laboratory or institute infrastructure, ensuring seamless integration with existing systems. The software should have minimal administrative and learning overhead to facilitate easy adoption and use.

Data Accessibility The interactive RWE should prioritize the use of existing file systems to avoid the need for databases that usually require expert knowledge to setup, maintain and extend.

Interactivity The interactive RWE should instantaneously allow users to remotely explore existing 2D/3D multi-modal images and create new image related data, such as image annotations, in an interactive manner. In addition, the interactive RWE should eliminate the need for large data transfers.

Scalability The interactive RWE should ensure scalability, allowing multiple users to work on the same resources simultaneously without encountering conflicts.

Maintenance To ensure long-term sustainability of the algorithms and workflows, the interactive RWE should be designed for easy maintenance, updates, and expansion. This includes the ability for non-specialized technical staff to install components. The interactive RWE should be adaptable to meet new developments and changing requirements.

Collaboration The interactive RWE should encourage collaboration by providing any number of interactive applications to multiple Data Users or Data Analysts.

4.3 Platform Architecture

The main goal is to facilitate functions that support the remote development process in the context of MSI. Since the development of new methodologies represents the most complex and demanding aspect of the platform, other functionalities, such as remote exploration, viewing utilities, and collaborative tasks, naturally extend from this foundation. These capabilities are designed to enhance the user experience, promote efficiency, and support collaborative efforts among researchers and developers working in different locations. This holistic approach ensures that the platform not only meets the core requirements of method development but also provides comprehensive tools for the broader scope of remote biomedical research activities. This section outlines the general architecture of the platform, as depicted in Figure 4.3 *Platform Architecture of the Interactive Remote Working Environment*. The two main components of the platform architecture are:

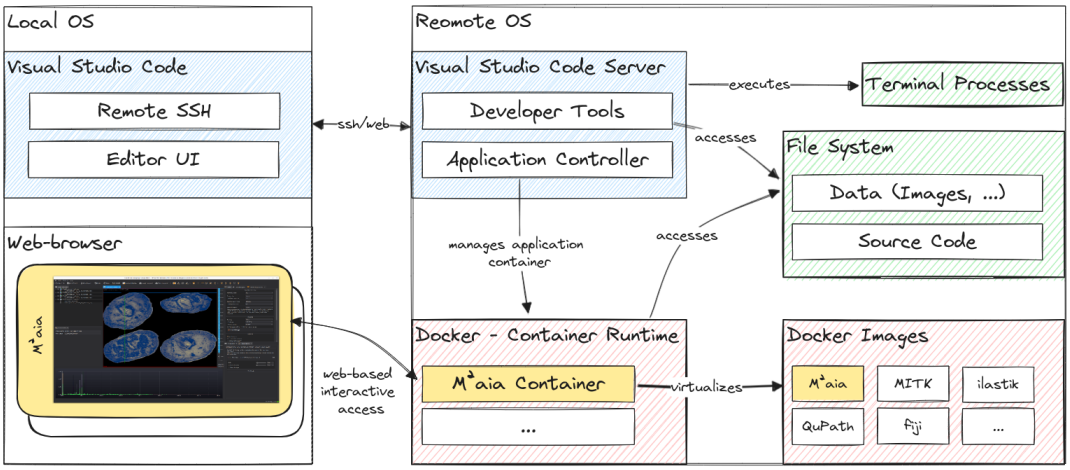

FIGURE 4.3. Interactive Remote Working Environment - Platform Architecture: On the Local Operating System (OS) VS Code and a web browser are required. On the Remote OS, VS Code Server and a Docker Host are required. Users can conveniently access a server by using the Remote SSH extension (or Remote Tunnels). All required extensions for application development, like specific programming language support and other features, are installed as Workspace Extensions on the Remote OS. The Application Controller extension provides convenient management of Applications Containers directly from within the VS Code's user interface. VS Code has the opportunity to access remote allocated data and manage Terminal Processes. Access to an interactive application is realized web browser-based by bringing remote hosted interactive desktop applications like M²aia, MITK, ImageJ, QuPath, ilastik and others, ie. community contribution, directly to the Local OS.

Remote Integrated Development Environments (IDE): the desired architecture includes a dual-interface approach consisting of a remote Integrated Development Environment (IDE) server and a corresponding IDE server interface on the local site. The server interface can be a web-service such as JupyterLab[145], RStudio[97] or a desktop client application such as Visual Studio Code (VS Code)[146]. This interface is central and primarily serves Data Analysts developing new methods by using the code editor. It also includes tools for managing data transfers and remote processing tasks. Visual Studio Code (VS Code) is chosen in the further conceptualization as the primary server interface for the local and server operating system because of its extensive support for remote development and the execution of tasks. It provides an efficient coding environment and integrates seamlessly with many differed programming languages and development tools. Besides the wide range of excellent IDE functionalities and supported programming languages, a whole set of community driven extensions for all kind of tasks exists. In this work we encourage the use of VS Code extensions supporting remote access to server resources.

The Remote Development extension pack is utilized to streamline the remote development process. This extension pack includes two extensions that facilitate access to centralized resources, namely the Remote SSH and Remote Tunnel extensions. Both extensions offer mapping of remotely located folder structures to the local system, thereby emulating a local development environment that leverages remote resources.

Docker-Based Application Container: Docker is a containerization platform that allows software developers to package their applications and dependencies into portable containers that can be easily deployed across different environments. These containers provide a consistent and reliable environment for running applications,

making it easier to develop, test, share, and deploy software. In the proposed concept Docker[112] is installed on the server side as application provider service (see section 2.5 *Containerization with Docker*).

The packaging of interactive applications in a Docker containerScherer *et al.*, (2020,)[2,147] is an important concept of this work and realizes the proposed capabilities of remote interactive access to image dataset. The design of these containers focuses on hosting a single interactive application that is accessible from outside the container using web-based communication. Packaging a single application within a container optimize both performance and resource utilization. The concept for the interactive application container includes three essential components:

- A Unix-based desktop environment, such as GNOME Shell, LXQt, LXDE, Xfce, or MATE, to facilitate the execution of interactive applications[148].
- An interactive desktop application specifically designed for visualizing and processing biomedical image data.
- A remote desktop service that enables access to the Unix desktop and its applications via a web browser.

An additional (optional) components in order to facilitating the user-interaction on the management level can be realized:

Application Controller: to facilitate the use of application containers, container management plug-ins are beneficial. For example, convenient access to an IDE-integrated utility to create, remove, observe, configure and share containers directly within the remote IDE's UI is a helpful tool for the Data Analyst. Alternatively, it would be possible to manually launch the application containers directly using the Docker Application Programming Interface (API), but this would make it much more difficult to manage the applications if it does not exist. A custom VS Code extension is implemented for the proposed platform architecture, that is based on VS Code, allowing for a streamlined management of remote applications. More details on can be found in section 4.5 *Application Controller*

In summary, this architecture is designed to provide a robust, scalable, and user-friendly environment for remote processing of biomedical image data using Docker, a remote IDE, and the optional proposed Application Controller, ultimately providing convenient access to the component of the RWE.

4.4 Remote Interactive Applications

This section describes the application containers that provide access to the interactive applications. These application containers are designed for rapid deployment and are particularly beneficial in development scenarios, as they provide direct interaction with imaging datasets while taking advantage of the full processing capabilities of CPUs and GPUs. At its core, access to containerized applications can be realized through Video Network Computing (VNC) capabilities that capture the desktop screen running inside the container. This setup can be accessed externally through an HTML5 compliant web interface. The concept of these interactive application containers is further elaborated in Figure 4.4 *Interactive Remote Working Environment - Application Container*.

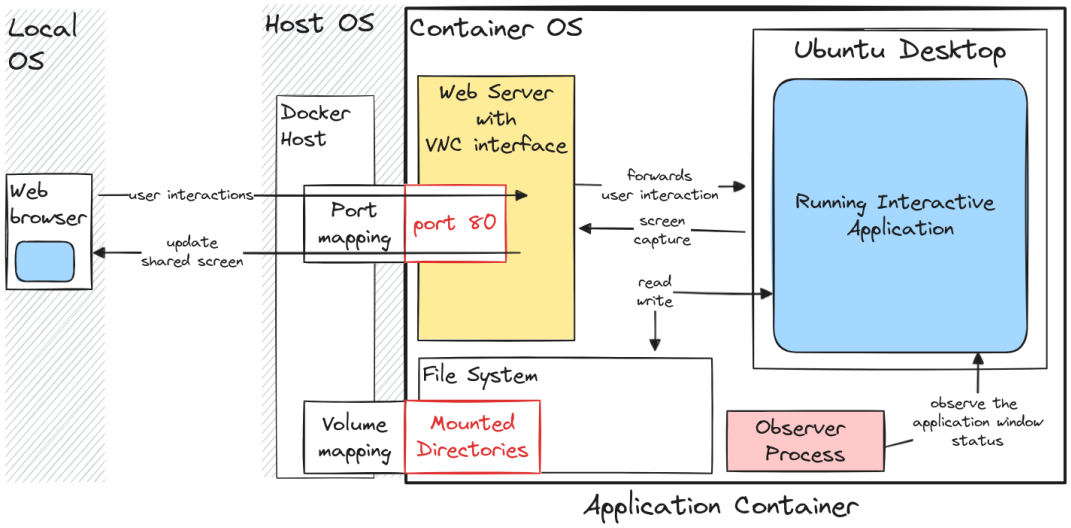

FIGURE 4.4. Interactive Remote Working Environment - Application Container: Docker is used to start an Application Container using port and volume mappings. The application container is providing an Ubuntu Desktop (e.g. LXDE/LXQt). At startup, the web-server and the VNC interface (yellow box) as well as the observer process (red box) are started as background processes. Additionally, the interactive desktop application (blue box) is started in full screen mode. The observer process (bash script) checks the window status and forces the container to shut down if the application crashes.

Application Images: a Docker-based image hierarchy is proposed to create new interactive applications. This is illustrated in Figure 4.5 *Hierarchy of Docker Images to Encapsulate Interactive Applications* and shows the dependencies of the Docker images used to build and package new applications. The base image `local/vnc-base` provides the desktop runtime environment and interactive web browser access[149]. A new application Docker image is created by inheriting from this base image. The new image uses instructions to provide all runtime dependencies and downloads/copies the image processing application into the Docker image. Installed applications are set up as the startup application of the desktop environment using a `supervisord`[i] config file and startup script.

[i]Supervisord is a process control system for UNIX-like operating systems that allows users to monitor and control processes on UNIX and Linux systems. It is commonly used to start, stop, and restart background daemon processes as defined in a configuration file. Example file on Github https://github.com/m2aia/m2aia-docker/blob/main/applications/m2aia/files/supervisord.conf; accessed April 2024

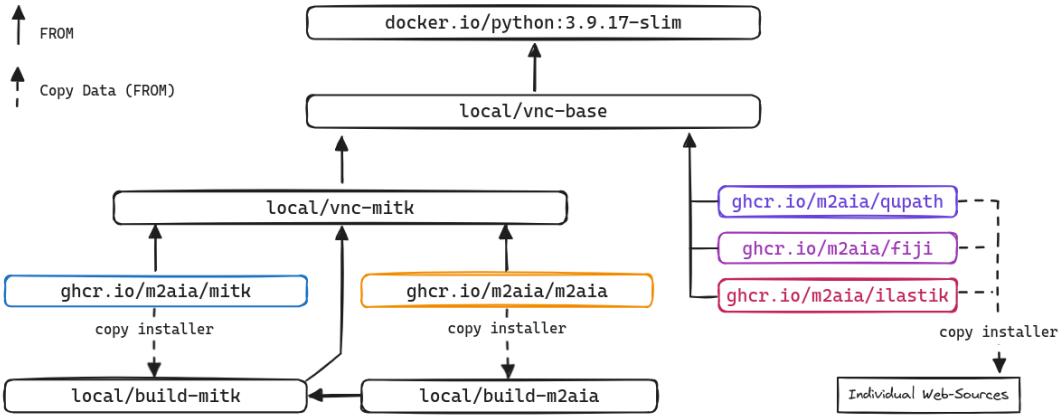

FIGURE 4.5. Hierarchy of Docker images used to encapsulate interactive applications - The base of the hierarchy is an Ubuntu based Python image in order to support pyM²aia by default. The openly accessible Docker registries `docker.io` and `ghcr.io` are used to publish the images. Local build images are marked with `local/`. The `local/m2aia/vnc-base` provides the Desktop environment and the VNC server. The `local/m2aia/vnc-mitk` provides all runtime dependencies for MITK and M²aia. In order to provide a application executable, it either can be download from a web-source as shown for `ghcr.io/m2aia/ilastik`, `ghcr.io/m2aia/qupath`, and `ghcr.io/m2aia/imagej`, or compiled by a build image such as highlighted by `local/build-mitk` adn `local/build-m2aia`. Colored images are made publicly accessible.

```
# Use the vnc-base image from the
FROM local/m2aia:no-vnc-base

# Add labels to identify the image and its version
LABEL IMAGE="image-processing-application"
LABEL VERSION="0.x.y"
RUN mkdir -p /src /app

RUN apt-get update && apt-get install dependency_a dependency_b

# Download the image-processing-application installation package from Github and extract it to the /new_app
    directory
RUN wget https://github.com/app/app/releases/download/v0.x.y/image-processing-application_0_x_y.tar.xz \
    -O /src/app.tar.xz && \
    tar -xvf /src/app.tar.xz --strip 1 -C /app/ && rm -rf /src/app.tar.xz

# Make the image-processing-application_0_x_y executable
RUN chmod +x /app/bin/image-processing-application_0_x_y

# Copy a shell script that starts the application to the Desktop folder and make it executable
COPY files/startapp.sh /root/Desktop/
RUN chmod 0777 /root/Desktop/startapp.sh

# Set the working directory to /data
WORKDIR /data

# Copy a supervisord config file that manages the application
COPY files/supervisord.conf /etc/supervisor/conf.d/
```

LISTING 4.1. A definition of a Docker image to create an interactive application image.

Start an Application Container: to start an application container, the command, shown in Listing 4.2 *Docker command to start an application container* can be executed using the Docker API. The application can be accessed by the container address using a web browser.

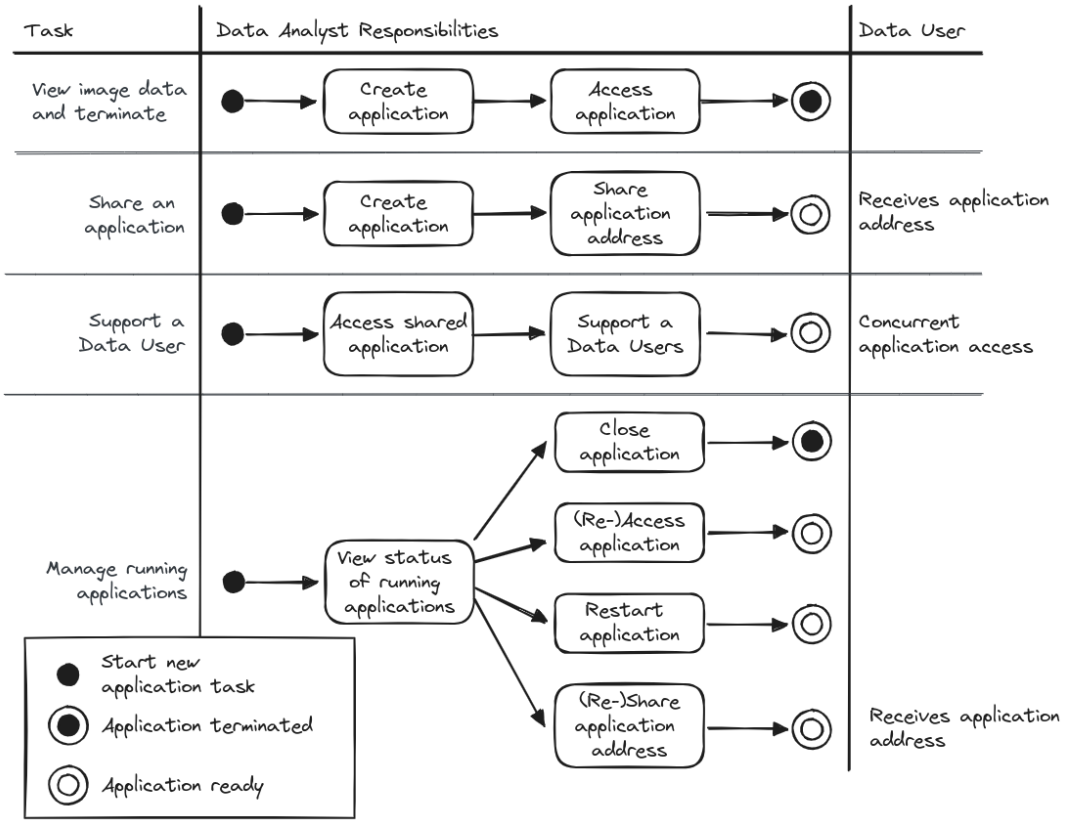

FIGURE 4.6. Data analyst tasks in the remote working environment. Tasks associated with Data Analysts (left column), which can result in the "finished" or "running" state (middle column) of an interactive application. Data Users can be part of a task when collaboration is focused, for example, by receiving an application address.

```
docker run --rm -d\
    -P \ #assign a random free port
    -e USER=${USER_NAME} \ # user name
    -e USERID=${USER_ID} \ # user id
    -e GROUPID=${USER_GROUP_ID} \ # users group id
    -e RESOLUTION=${RESOLUTION} \ # screen resolution
    ${VOLUME_MAPPINGS} \ # e.g -v /my/data/dir:/data/dir/in/container
    ${IMAGE_NAME} # e.g ghcr.io/m2aia/m2aia:v2023.10
```

LISTING 4.2. Docker command to start an application container.

4.5 Application Controller

The management of Docker-hosted interactive applications includes the creation, status tracking, and configuration of different application containers. Tasks in order to provide a flexible environment for remote image processing with interactive applications are shown in Figure 4.6 *Tasks of Data Analysts in the Remote Working Environment*. The controller should provide a streamlined workflow specifically designed for managing application containers within the remote IDE and offer utilities to control, monitor, and adjust Docker-based interactive applications directly from within the interface. The objective of these interactions is to enhance productivity and facilitate the utilization of the software ecosystem for Data Analysts, both during the development process and subsequent analysis phase. In this work, a newly developed

VS Code extension named vcM²aia is introduced following these principles[i].

[i]https://m2aia.de; accessed April 2024.

Chapter 5

Results

This chapter presents the results of the different research projects presented in the chapters 3 and 4. The chapter consists of six parts. The first part in section 5.1 corresponds to the datasets used for the experiments. The second part corresponds to the results obtained for the concepts of the interactive 2D/3D MSI data processing application in section 5.2. The third part demonstrates the processing of MSI datasets with Python with a focus on deep learning in section 5.3. The fourth part demonstrates several different integrations of third-party methods in section 5.4. The fifth part includes demonstrations of the remote working capabilities in section 5.5. The sixth and final part contains the data and code availability statements.

5.1 Materials

5.1.1 Lipid/Peptide 3D APP NL-G-F Mouse Brain

To demonstrate the capabilities of M²aia a lipid 3D and a peptide 3D MSI dataset is made publicly available[i]. The dataset consists of 10 consecutive sections of brain tissue taken from an APP NL-G-F mouse model. Briefly, the sample was cut with a thickness of 10 µm and sections were placed on a single Bruker indium-tin oxide slide. Subsequently, lipid and peptide MALDI-TOF-MSI acquisitions were made. In between the data acquisition for lipids and peptides, the matrix and most of the lipids remaining on the tissue sections were washed away before the peptide acquisition protocol was applied. The lipid and peptide datasets share a common lateral resolution of 20 µm and a spot size of 20 x 20 µm. The dataset and this description is published in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0). Unless otherwise described, this dataset will be further referred to as *Mouse Brain Dataset*.

5.1.2 Adult Earthworm (L. rubellus)

The second dataset comprises four sections (sections 1-4) of an adult earthworm (L. rubellus) and was published by Geier et al.[26]. Key imaging details from the original publication are summarized as follows: MALDI-MSI was performed using a 7 mg·mL^{-1} α-cyano-4-hydroxycinnamic acid matrix in a 70:30 acetonitrile/water mixture containing 0.2% trifluoroacetic acid, applied with SunCollect's automated spray-coating system. Key parameters included a capillary z-distance of 25 mm, compressed air pressure of 2 bar, and flow rates varying from 15 μL·min^{-1} for the first layer to 20 μL·min^{-1} for layers 2-8. The imaging was conducted using an Autoflex speed LRF MALDI-TOF with a smartbeam-II 1 kHz laser, a 25 μm spot size, and a "random walk" pattern. Acquisition involved 500 shots per sampling point (100 shots

[i] http://gigadb.org/dataset/100909; accessed April 2024

per location within the spot), covering a mass detection range of m/z 100 to 1,280 with an accuracy of 200 ppm. Unless otherwise described, this dataset will be further reffed to as *Earthworm Dataset*. The dataset is publicly available in the MetaboLights repository [MTBLS2639].

5.1.3 3D Reference Datasets

The study published by Oetjen *et al.* (2015)[33] focuses on providing multiple benchmark datasets for 3D MALDI-MSI. These datasets are intended to stimulate computational research in 3D MSI. The reference datasets are publicly available in the MetaboLights repository [MTBLS176]. The individual datasets are summarized in the following list:

- 3D Mouse Kidney Dataset: this dataset comprises 75 sections from a central part of a mouse kidney that was fixed and embedded in paraffin. Spectra were acquired from the 3.5 µm thick sections. This process resulted in a total of 1,362,830 spectra, with each spectrum containing 7,680 data points. The data were preprocessed with Gaussian spectral smoothing and baseline reduction, and then registered using SCiLS Lab software to reconstruct the original 3D structure of the kidney and exported to imzML. The total size of the data set is 44.2 GB, of which the `.imzML` file is 2.4 GB.
- 3D Mouse Pancreas Dataset: the pancreas from a C57BL/6 mouse was fixed and embedded in paraffin. Spectra from 29 sections (5 µm thick) were acquired. The dataset includes 497,225 spectra with 13,312 data points per spectrum. Data were processed and visualized using SCiLS Lab software, with image registration and conversion into imzML format for further analysis. The total size of the data set is 27.3 GB, of which the `.imzML` file is 0.86 GB.
- 3D Human Oral Squamous Cell Carcinoma (OSCC) Dataset: Tissue sections (10 µm thick) from a patient with OSCC. 58 sections were analyzed, producing 828,558 spectra with 7,680 data points per spectrum. Spectral data were preprocessed and registered using SCiLS Lab software to produce a 3D volume, then exported to imzML format. The total size of the data set is 26.8 GB, of which the `.imzML` file is 1.4 GB.
- Microbe Interaction Time Course Data: the time course data involves analyzing metabolic exchanges between Streptomyces coelicolor A3(2) and Bacillus subtilis PY79 over time. Spectra were acquired at three time points (days 1, 4, and 8 post-inoculation). The dataset includes 17,672 spectra with 40,299 data points per spectrum, which were processed and registered using SCiLS Lab software to create 3D volumes. The total size of the data set is 2.9 GB, of which the `.imzML` file is 0.03 GB.

3D reconstruction of the image volume was realized with the so-called user-guided rigid registration. These datasets are made available in imzML format, facilitating their use in developing and testing new computational methods for 3D imaging MS.

5.2 Interactive 2D/3D MSI Data Processing Application

The results of the concepts introduced in chapter 3 *Interactive Multi-Modal 2D/3D MSI Data Analysis* are presented in this chapter and is structured as follows: The results of the data import are presented in subsection 5.2.1. Based on the ability to import MSI data, features required for general interactions with MSI data are presented

in subsection 5.2.2. The next three subsections present use-cases for biomarker identification in subsection 5.2.3, multi-modal 3D reconstruction in subsection 5.2.4, and multi-modal image fusion in subsection 5.2.5. All features and use-cases shown are realized in the in this work realized stand-alone desktop application called *Mass Spectrometry Imaging Applications for Interactive Analysis in MITK (M²aia)*[116].

5.2.1 Data Import and Performance

The importation of data and the subsequent performance of the system are two key areas of focus. In this subsection the concepts (introduced in section 3.2 *Concepts for MSI Data Processing*) for data import are presented. The following two paragraphs describe the results of data access experiments to metadata and spectral data stored in imzML format.

Access to Metadata: The metadata contained in imzML files is essential for detailing a wide range of experimental parameters and conditions. This metadata encompasses various aspects, including the specification of the instrumentation used, the parameters set during the data acquisition process, and the processing steps performed during the experiment. Accessing these detailed data points is made straightforward through the imzML reader provided by M²aia.

Metadata is accessed in the form of a dictionary, making it easy to navigate and retrieve specific pieces of information. For example, users can quickly extract details about the type of mass spectrometer used, the settings applied during data acquisition, and any preprocessing steps that were executed. This capability is critical for researchers who need to ensure that their analyses are transparent and reproducible, as they can precisely document the experimental conditions under which their data was generated.

In C++, metadata access can be realized using the concepts and classes introduced in subsection 3.2.1 *Hyperspectral data import*. These classes provide a structured approach to handling metadata, allowing developers to integrate metadata retrieval seamlessly into their data processing pipelines. The C++ implementation ensures high performance and is suitable for applications requiring efficient data handling.

For scripted metadata retrieval applications, such as those involving large data collections, the pyM²aia-based access is described in detail in subsection 5.3.2 *Metadata Extraction*. PyM²aia offers a user-friendly interface for scripting in Python, enabling researchers to automate the extraction and analysis of metadata across numerous datasets. This automation is particularly beneficial when dealing with high-throughput experiments, where manually accessing metadata for each dataset would be impractical.

By leveraging pyM²aia, researchers can create scripts that systematically retrieve and analyze metadata, facilitating large-scale studies and meta-analyses. This approach supports the organized retrieval of information regarding the experimental setup, ensuring that all relevant details are captured and can be easily referenced in future studies. Ultimately, the ability to efficiently access and utilize metadata enhances the reproducibility and transparency of scientific research.

Import of imzML MSI Datasets: an important aspect of M²aia is to provide a fast and memory-efficient access to one or multiple 2D/3D MSI datasets at the same time. To demonstrate the capabilities of M²aia in handling imzML MSI datasets, performance metrics are presented in Table 5.1 *Timing experiments on reference dataset*.

The metrics utilize publicly available 3D reference datasets from Oetjen *et al.*[33], accessible through the MetaboLights repository [MTBLS176]. The performance evaluation, including processing times and memory usage, was averaged over three separate runs to ensure reliable data. This assessment was performed on two distinct system configurations: a mobile setup with an Intel® Core™ i7-8750H CPU and a desktop setup featuring an AMD® Ryzen 9 5900x CPU. The detailed processing included total-ion-count (TIC) normalization. This paragraph has been published in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

TABLE 5.1. Timing experiments on mobile and desktop systems using 3D reference data. The table lists the file names, number of spectra, depth of a spectrum and the average time in seconds and memory usage, for three manually repeated runs. Applied signal processing includes total-ion-count (TIC) normalization. Data generated using the 3D reference datasets published by Oetjen *et al.*[33] and available in the MetaboLights repository [MTBLS176]. System configuration A: mobile PC, Intel® Core™ i7-8750H CPU @ 2.20GHz 6-core processor, 16 GB physical memory and SSD. System configuration B: desktop PC, AMD® Ryzen 9 5900x CPU @ @ 3.7GHz 12-core processor, 32 GB physical memory and M.2 SSD. Table in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0)

File Name	System	Size	Spectra/Depth	Parsing metadata	Initialization	Create ion-image	RAM usage
3D Mouse Kidney	A	44.2GB	1362830/7671	24.9s	26.27s	10.9s	435.1MB
3D Mouse Kidney	B	44.2GB	1362830/7671	5.7s	2.1s	0.6s	435.1MB
3D Mouse Pancreas	A	27.3GB	497225/13297	9.4s	13.14s	2.8s	292.4MB
3D OSCC	A	26.8GB	828558/7665	14.9s	13.4s	4.04s	323.5MB
Microbe Interaction 3D Timecourse	A	2.9GB	17672/40299	0.4s	0.9s	0.08s	231.8MB

5.2.2 Individual Features for MSI Data Exploration and Processing

The concepts of data import (see subsection 3.2.1 *Hyperspectral data import*), signal processing (see subsection 3.2.2 *Signal Processing*), peak picking (see subsection 3.2.4 *Peak picking*), and data compression (see subsection 3.2.5 *Data Compression*) have been integrated into M²aia and will be demonstrated in the following sections. Each paragraph refers to one or more of the methods presented there, defining a comprehensive set of features for MSI data exploration and processing. Parts of this section have been published in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

User interface: The user interface was designed to facilitate easy navigation and interaction for users conducting MSI data analysis. The primary goal was to create an intuitive, responsive, and visually appealing interface that enhances user experience. The user interface has a clear, modular layout and is orientated towards the application design of the MITK Workbench. It is divided into three columns. The left column contains elements for the data management (Data Manager view), the center column displays the main content (Standard Multi Widget view with render windows for displaying image data in 2D and 3D), and the right column provides views for processing of MSI data. This layout is dynamic an can be rearranged according to the needs. Views can be detached from the main window. The UI was built using MITK's concepts for creating new views and perspectives. M²aia is complied for multiple operating systems, including Windows- and Unix systems. In Figure 5.1 *Graphical user-interface of M²aia*, the default startup MSI processing perspective is shown on Ubuntu 22.04, after four MSI datasets were loaded.

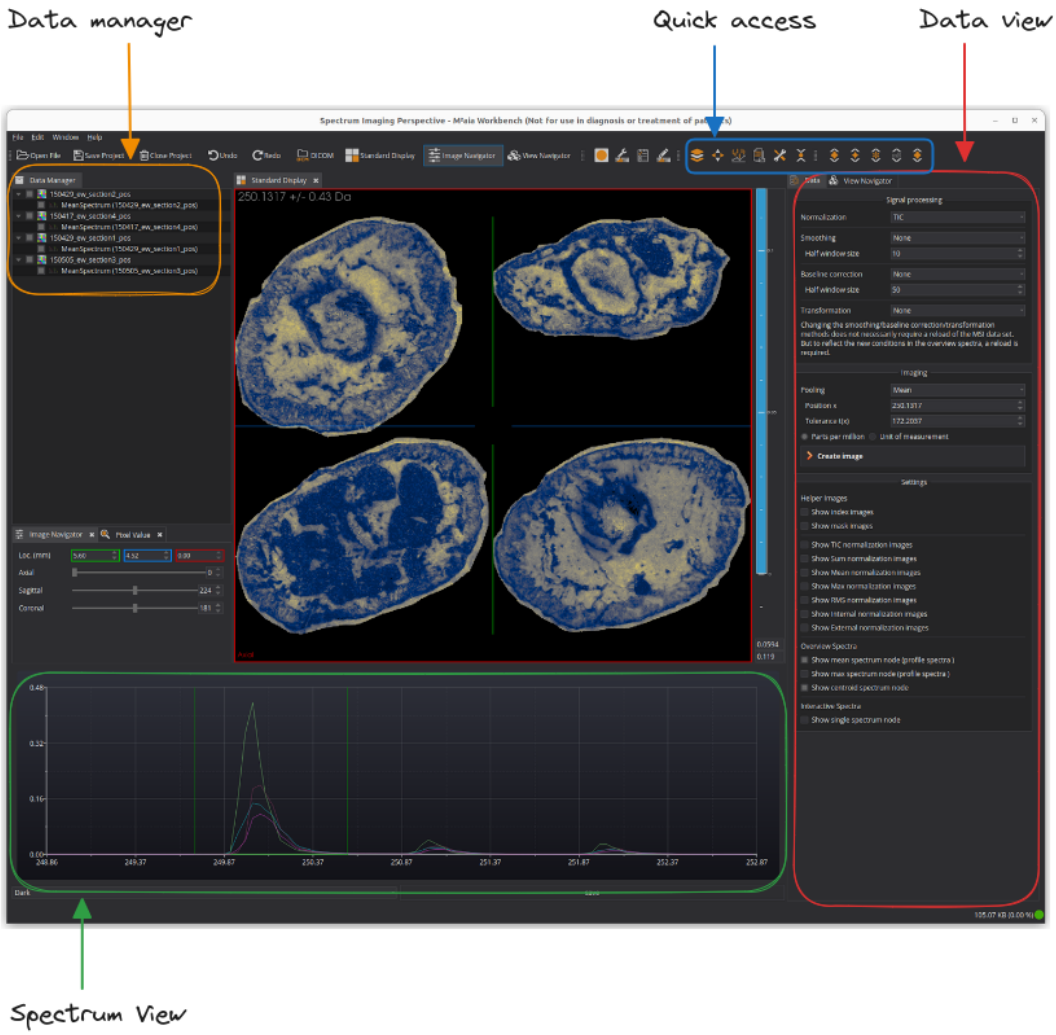

FIGURE 5.1. The graphical user interface of M²aia introduces MSI processing utilities and interactive workflows for 3D reconstruction and multimodal image registration. Rendered images show four slices of the *Earthworm Dataset* (see subsection 5.1.2 *Adult Earthworm (L. rubellus)*) published by Geier et al.[26]. Ion-images show intensities of m/z 250.13 ± 0.4 Da. The data view is shown on the right, the data manager on the left, and the spectrum view on the bottom (showing the average overview spectrum of all four MSI datasets).

Segmentation Tools: M²aia provides access to MITK's segmentation utilities. Segmentations are used to annotate structures in the data. In Figure 5.2 *Utilizing MITK's Image Segmentation View* a manual segmentation was created using the 2D tool. The ability to use the mature segmentation utilities of MITK for the new MSI based images enables advanced integrated workflows. The annotations created can be utilized for analysis within M²aia, thereby providing an indispensable tool for image-based workflows. This will become even more apparent with the integration of MSI-based methods into the interactive context of M²aia, as described below. This is related to concepts proposed in this thesis, such as the Python-based deep-learning concepts, which can benefit from annotations to realize supervised tasks (see [chapter/results:pym2aia-example-VII]). Thus, segmentations generated by integrated automatic methods such as k-means clustering or the hotspot/coldspot segmentations generated by MoleculaR[150] (subsection 5.4.5 *moleculaR - Collective Projections of Metabolites*) are fully compatible with MTIK's segmentation view, demonstrating the integrability of new MSI-based concepts with the existing segmentation capabilities of the MITK framework.

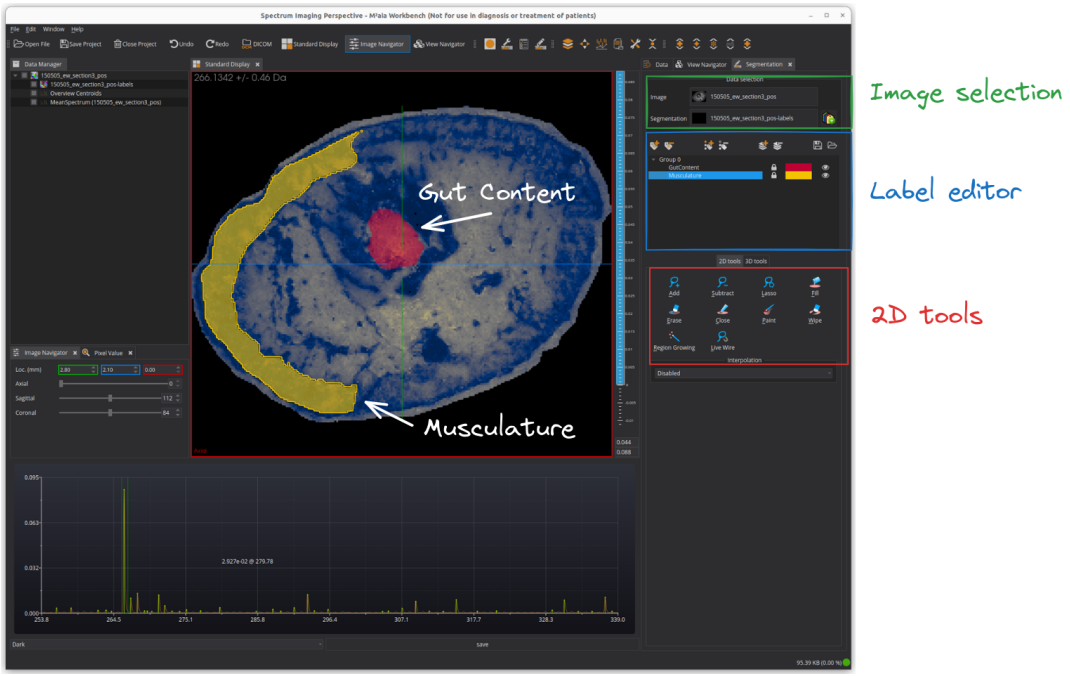

FIGURE 5.2. Illustration of M²aia showing the usage of MITK's segmentation utilities for MSI datasets. The segmentation view is on the right with image selection (green box), label editor (blue box), and tool selection (red box). MSI dataset shows an ion-image at m/z 266.13 ± 0.4 Da of slice 3 of the *Earthworm Dataset*[26] (see subsection 5.1.2 *Adult Earthworm (L. rubellus)*) .

Data Visualization and Interaction: all images loaded into M²aia are integrated within a unified world coordinate system that defines a virtual environment. This virtual space is accessible through several rendering windows, each presenting distinct planar slices of the virtual environment, a technique known as multi-planar reconstructions (see section 2.4 *Medical Imaging Interaction Toolkit*). Volume visualizations can be realized by using approperiate features of MITK. This is shown for 3D MSI data as a result in subsection 5.2.4 *Use-case: Multi-modal 3D Image Reconstruction* in Figure 5.10 *Visualization of 3D MSI datasets*.

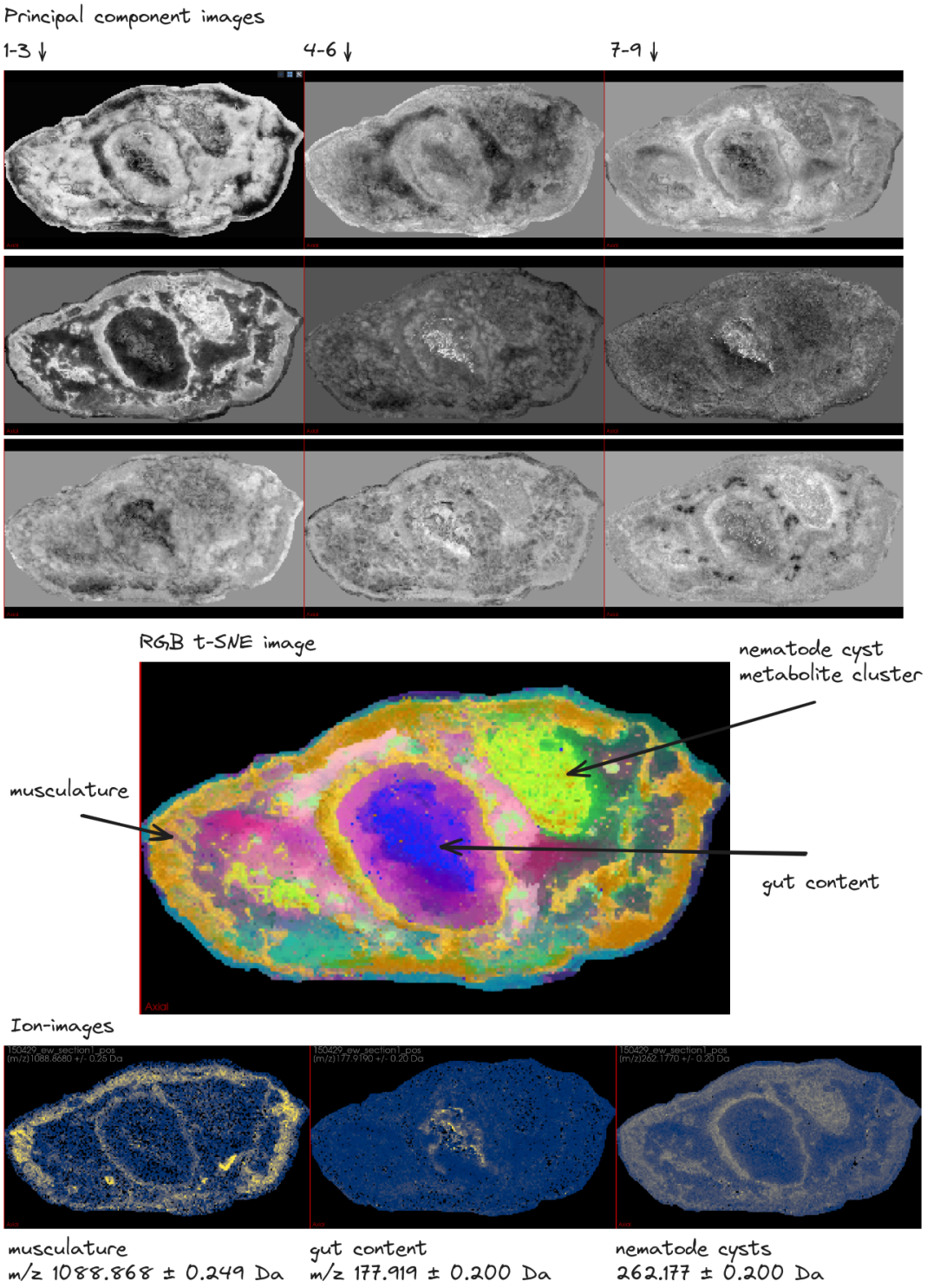

FIGURE 5.3. Visual demonstration of dimensionality reduction methods PCA and t-SNE. Input for all methods is an MSI dataset and peak picking to generate a list of centroids. The data used in this figure are generated using the Section 1 of the *Earthworm Dataset*[26] (see subsection 5.1.2 *Adult Earthworm (L. rubellus)*) . Local maxima peak picking was used (SNR=3; hws=5) to reduce the profile overview spectrum to 1324 centroids. Ion-images were generated with a tolerance of 175 pmm,TIC normalization, maximum pooling, and square root intensity transformation. The first 9 of 15 principal component images of the PCA are shown. The t-SNE (perplexity=5; iterations=200; theta=0.5; shrink-factor=1) based on the PCA reduction is shown in the middle. Reference ion-images for individual m/z values as published by Geier *et al.* (2021)[26] for musculature, gut content, and nematode cysts are shown at the bottom.

Data Compression: data compression methods (in subsection 3.2.5 *Data Compression*) include PCA, t-SNE and k-means clustering. All methods are made accessible within the implemented DataCompression view. The proposed concepts for the conversion of MSI data into ion-images (see subsection 3.2.3 *Ion Image Generation*) based on centroids generated with the provided peak picking (see subsection 3.2.4 *Peak picking*) is able to provide the respective data compressed representations of MSI datasets. This is illustrated for the PAC and t-SNE methods in Figure 5.3 *Data compression methods in M²aia*.

Image-based Registration Utilities: The image-based registration has been realized in an independent mitk project called mitk-elastix. It provides easy-to-use elements to implement the registration in the M²aia user interface. This includes data import and export functions as well as utilities for running the elastix command-line tools. These elements were used to implement the subsection 5.2.4 *Use-case: Multi-modal 3D Image Reconstruction* and subsection 5.2.5 *Use-case: Semi-automatic multi-modal image registration* use-cases. For the evaluation of the results in the context of image-based recording with MSI data, additional tools were made available, which are described in the next section.

Image-based Registration Evaluation: to evaluate image-based registrations, fused results can be evaluated using the MatchPoint[107] utilities of the MITK framework. These tools provide a wide range of subjective evaluation strategies that can be applied to visually assess the quality of a registration. The generated ion-images (see subsection 3.2.3 *Ion Image Generation*) were used to execute multi-modal registration of two MSI datasets of the *Mouse Brain Dataset*[128] (see subsection 5.1.1 *Lipid/Peptide 3D APP NL-G-F Mouse Brain*) . The MatchPoint tools were used to demonstrate the evaluation within the context of MSI data processing, illustrated in Figure 5.4 *Visual Evaluation of Image Registration Results*. Beside the qualitative assessment, quantitative evaluation could be realized using the point set interaction tools of MITK to place landmarks within both modalities and evaluate the TRE. This is shown from an interactive perspective in Figure 5.5 *Target registration error (TRE)*. Quantitative evaluation of the proposed registration strategies is part of subsection 5.2.4 *Use-case: Multi-modal 3D Image Reconstruction*. In both figures ion-images were generated using the TIC normalization and maximum pooling.

5.2.3 Use-case: Biomarker Identification

This use-case was published in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

A publicly available N-linked glycan MALDI-TOF dataset[151,152] is re-analyzed to demonstrate the applicability for biomarker identification task in M²aia. The dataset is available in the PRIDE repository with accession code PXD009808. The dataset is used to examine an automated sample preparation approach for MALDI-TOF/TOF imaging of N-linked glycans on formalin-fixed paraffin-embedded (FFPE) murine kidney tissue. PNGase F was printed on two FFPE kidney sections to release N-linked glycans from proteins. A part of the third kidney was covered with N-glycan calibrants and another part with buffer to serve as a control. Imaging was performed with a spatial resolution of 100 µm. Using M²aia, three datasets (PNG1, PNG2 and control; in total approx. 6.4 GB; skipping the calibrant area) were loaded

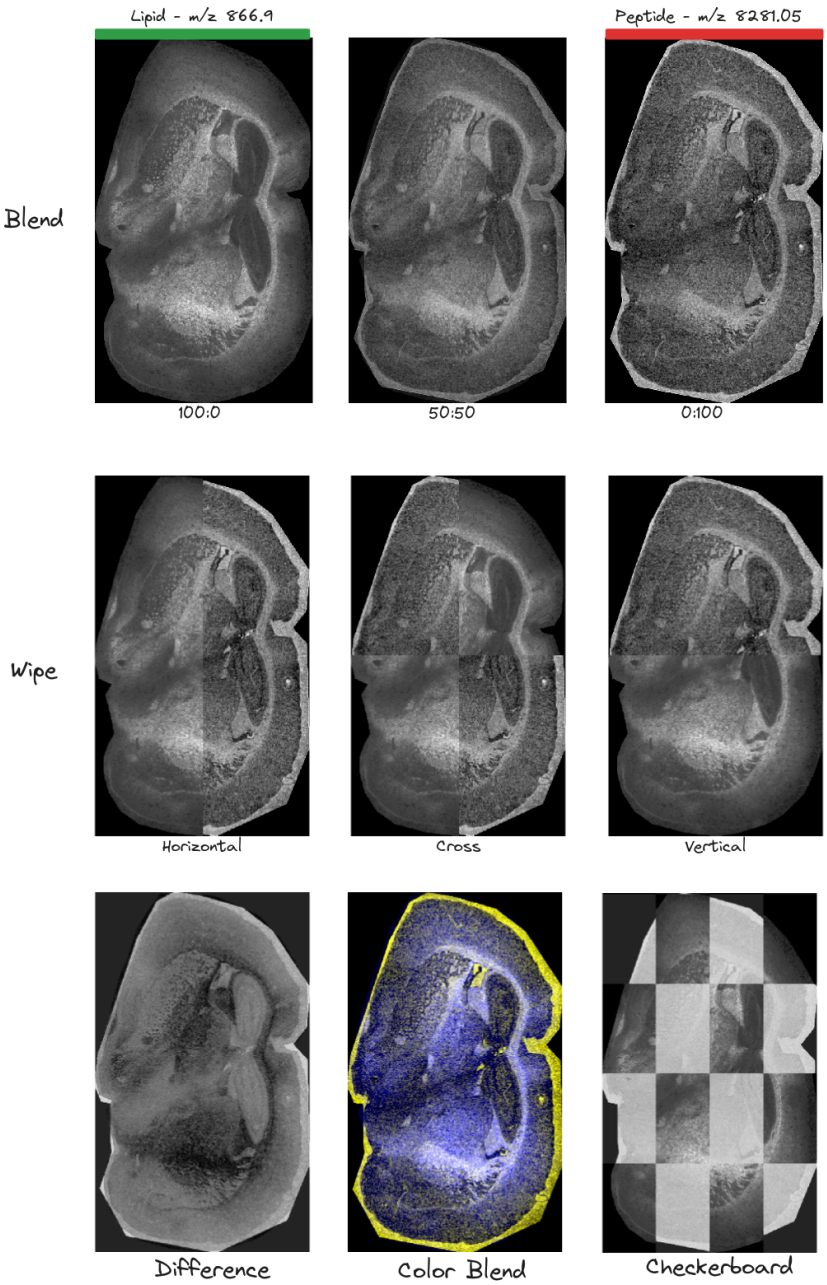

FIGURE 5.4. Visual evaluation of registration results offered by MITK. Multi-modal input ion-images of the *Mouse Brain Dataset*[128] (see subsection 5.1.1 *Lipid/Peptide 3D APP NL-G-F Mouse Brain*) . The input images for the registration are: red peptide MALDI MSI at m/z 8281 and green lipid MALDI MSI at m/z 866.9 from two different MALDI MSI acquisitions of the same sample.

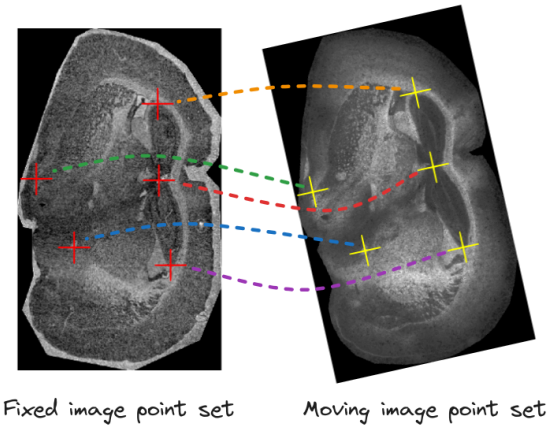

FIGURE 5.5. Set of pair-wise defined points for the quantification of the Target Registration Error (TRE). Points can be set interactively using the point set interaction view of MITK. Multi-modal input ion-images of the *Mouse Brain Dataset*[128] (see subsection 5.1.1 *Lipid/Peptide 3D APP NL-G-F Mouse Brain*) : right peptide MALDI MSI at m/z 8281 and left lipid MALDI MSI at m/z 866.9 from two different MALDI MSI acquisitions.

and TIC normalization, Savitzky-Golay smoothing and Top-Hat baseline correction were applied (Figure 5.6 *Biomarker Identification*, Data preparation). Peak picking with monoisotopic peak identification was applied to the mean spectrum of each image, respectively. The peak results of the datasets were combined into a common peak list. Peak binning was applied to remove duplicates, resulting in a list of 107 m/z (candidate) peaks (Figure 5.6 *Biomarker Identification*, Feature extraction). To demonstrate how M²aia is used in combination with other tools, the processed data were exported as a single imzML file to continue the processing with Cardinal (2.6.0)[73]. Providing the list of common peak features during the export process allows to store the data in continuous centroid format. Using Cardinal, the two PNGase F treated kidney tissue sections are compared with the control tissue section for the identification of discriminant m/z candidates that are potentially related to N-linked glycans. N-linked glycan m/z candidates were selected by the supervised spatial shrunken centroids (SSC) algorithm[153,154] (Figure 5.6 *Biomarker Identification*, Analysis). Therefore, all pixels were separated into the classes "treated" (for PNG1 and PNG2) or "untreated" (for Control). By mapping the *treated* features selected by the SSC to the original publication of Gustafsson *et al.*[151], 16 N-linked glycan related m/z candidates were identified, as listed in Table 5.2 *Potential m/z-candidates related to N-linked glycans*. PCA images including the first three principal components and a t-SNE image (Figure 5.7 *Dimensionality Reduction*) were calculated based on the common peak list in M²aia.

For reproducibility purposes, protocols of the interactive steps from loading to exporting[155] and for dimensionality reduction[156] are available. The R-based processing of the intermediate results is available as a CodeOcean capsule[157,158]. An additional CodeOcean capsule implements the described workflow as a command-line application[159,160], demonstrating the possibility to develop M²aia-based applications for batch-processing and porting them to a server infrastructure.

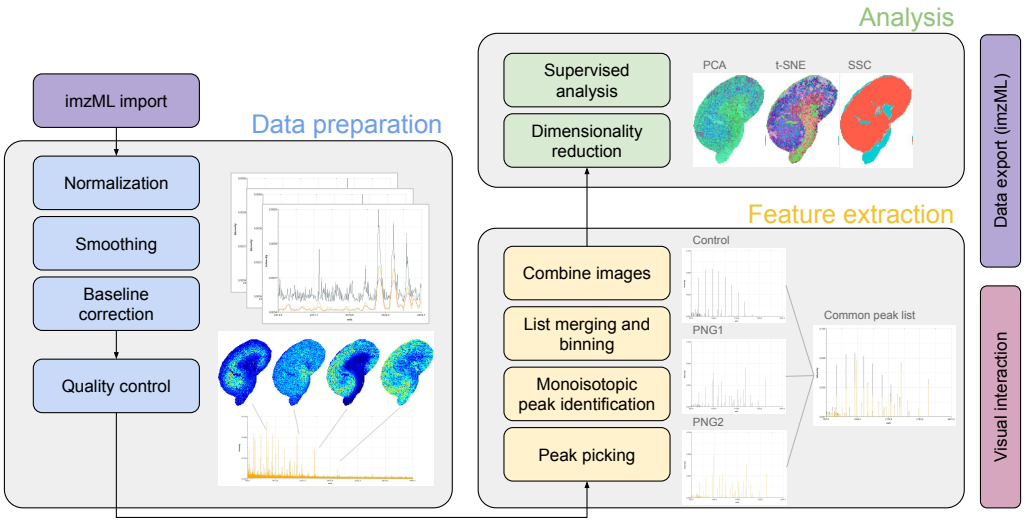

FIGURE 5.6. Typical steps to transform an MSI dataset into discriminating information in M²aia, exemplified by images from the re-analysis process of a N-glycan MALDI-TOF MSI dataset. Illustration in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

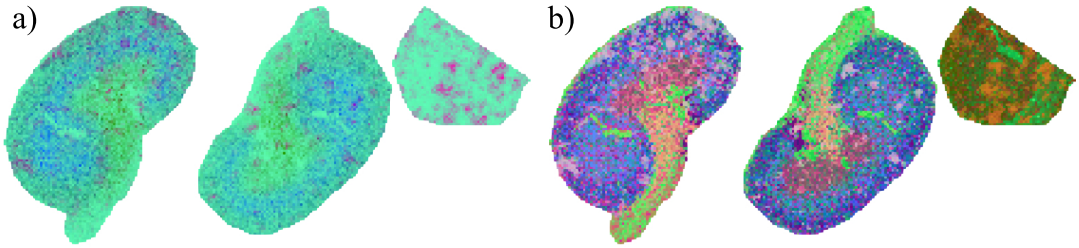

FIGURE 5.7. Results of two dimensionality reduction methods after performing the pipeline shown in Figure 5.6 *Biomarker Identification*. In a) the three principal components with the largest eigenvalues of a PCA and in b) results of a t-SNE with a target dimension of three are shown. Illustration in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

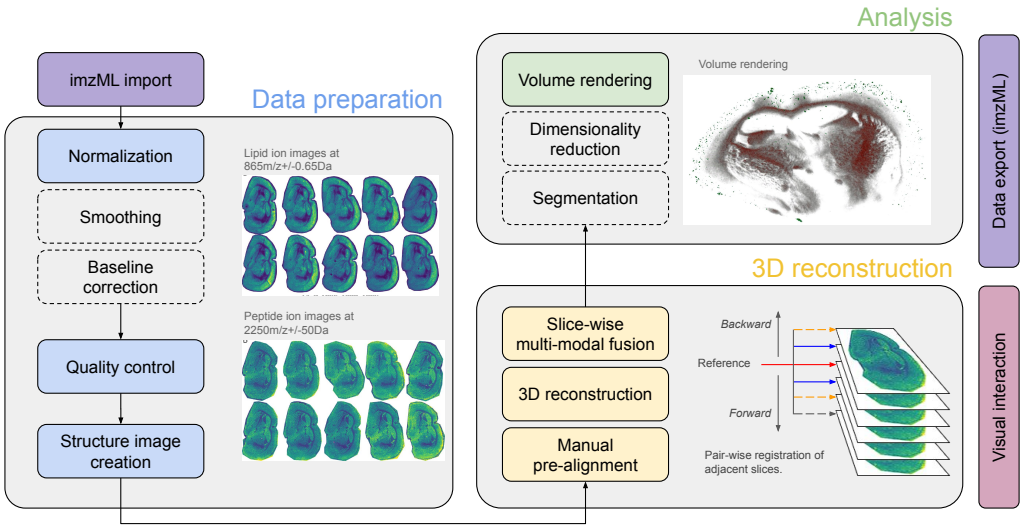

FIGURE 5.8. Steps for 3D reconstruction of consecutive MSI image slices in M²aia, exemplified by images from the 3D reconstruction and registration of the publicly available MALDI-TOF lipid and peptide dataset. Dashed boxes are possible additional processing steps that were not applied to the data shown. Illustration in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0)

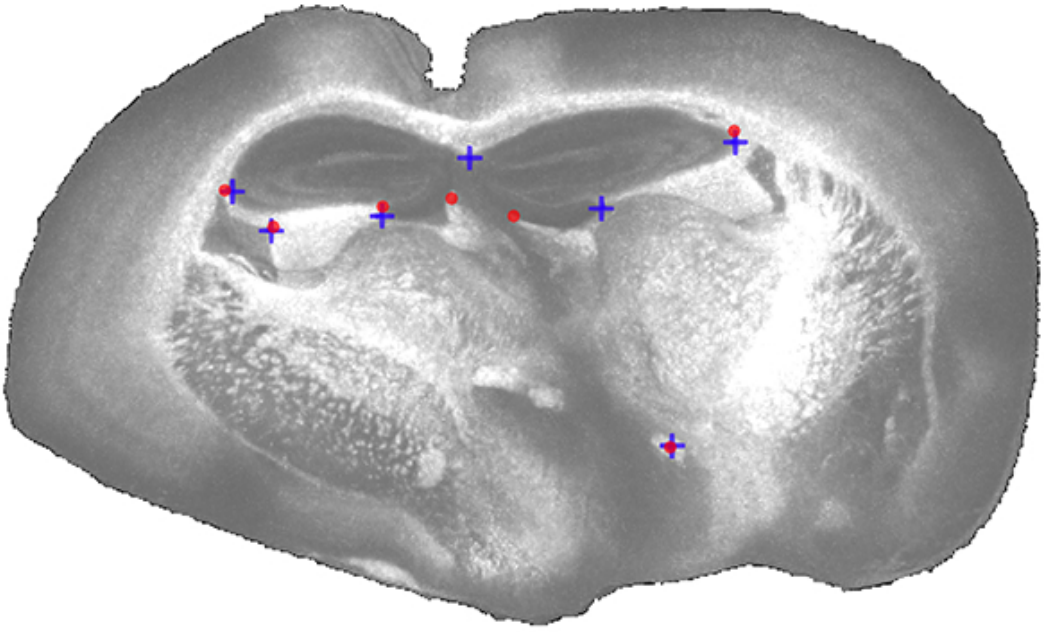

FIGURE 5.9. Maximum intensity projection of the multi-modal 3D reconstructed dataset at m/z 865±0.65 Da. Peptide (blue crosses) and lipid (red circles) reference points are shown for the mid-slice of the stack. Illustration in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

TABLE 5.2. Potential m/z-candidates related to N-linked glycans. By the re-analysis of the data published by Gustafsson *et al.*[151] in M²aia, a set of 16 discriminating m/z features (col. 2) was identified and mapped to the LC-MS/MS experiment Gustafsson *et al.*[151] (col. 6) for the treated kidney sections. Errors are listed for M²aia (col. 4) and Föll et al.[76] (col. 5) for comparison. Compositions (col. 7) as identified by Gustafsson *et al.*[151] of the corresponding m/z features. t-Statistics of the supervised spatial shrunken centroids algorithm (col. 3). Identifier (id, col. 1) for sorted features by descending t-statistics. Hex: Hexose, Man: Mannose, GlcNAc: N-Acetyl-D-glucosamine, HexNAc: N-Acetyl-D-hexosamine. Table in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

Id	m/z	t-Statistics	Error (ppm)	Error (ppm) Föll *et al.*[76]	LC-MS/MS M+NA+	Composition
1	1257.4730	63.79	50	51	1257.41	$(Hex)_2+(Man)_3(GlcNAc)_2$
2	1419.5177	60.62	33	59	1419.47	$(Hex)_3+(Man)_3(GlcNAc)_2$
3	1743.6281	54.50	33	67	1743.57	$(Hex)_5+(Man)_3(GlcNAc)_2$
4	1905.6748	50.93	23	30	1905.63	$(Hex)_6+(Man)_3(GlcNAc)_2$
5	1581.5697	50.58	25	61	1581.53	$(Hex)_4+(Man)_3(GlcNAc)_2$
6	2304.8962	47.12	28	36	2304.83	$(Hex)_2(HexNAc)_3(deoxyhexose)_3+(Man)_3(GlcNAc)_2$
7	1850.7140	46.86	34	34	1850.65	$(Hex)_1(HexNAc)_3(deoxyhexose)_1+(Man)_3(GlcNAc)_2$
8	1809.6975	45.44	37	52	1809.63	$(Hex)_2(HexNAc)_2(deoxyhexose)_1+(Man)_3(GlcNAc)_2$
9	2158.8425	38.62	33	54	2158.77	$(Hex)_2(HexNAc)_3(deoxyhexose)_2+(Man)_3(GlcNAc)_2$
10	1663.6324	35.92	37	58	1663.57	$(Hex)_2(HexNAc)_2+(Man)_3(GlcNAc)_2$
11	1485.5967	33.83	44	63	1485.53	$(HexNAc)_2(deoxyhexose)_1+(Man)_3(GlcNAc)_2$
12	1688.6586	31.59	28	62	1688.61	$(HexNAc)_3(deoxyhexose)_1+(Man)_3(GlcNAc)_2$
13	2012.7717	26.88	30	37	2012.71	$(Hex)_2(HexNAc)_3(deoxyhexose)_1+(Man)_3(GlcNAc)_2$
14	1647.6444	26.83	45	-	1647.57	$(Hex)_1(HexNAc)_2(deoxyhexose)_1+(Man)_3(GlcNAc)_2$
15	2816.1882	24.31	63	63	2816.01	$(Hex)_3(HexNAc)_4(deoxyhexose)_1+(Man)_3(GlcNAc)_2$
16	2067.7292	19.00	28	43	2067.67	$(Hex)_7+(Man)_3(GlcNAc)_2$

5.2.4 Use-case: Multi-modal 3D Image Reconstruction

This use-case was published in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

The objective is to demonstrate the applicability of M²aia for mono- and multi-modal 3D image reconstructions by showing how to embed the peptide information into the lipid structural context in three dimensions. The dataset in this use-case consists of 10 consecutive brain slices of the *Mouse Brain Dataset*[128] (see subsection 5.1.1 *Lipid/Peptide 3D APP NL-G-F Mouse Brain*), imaging both lipid and peptide features (in total approx. 80 GB in size). To demonstrate mono-modal 3D reconstruction, all 10 slices of the lipid and the peptide datasets were loaded into M²aia, respectively, and lipid datasets used for slice-wise reconstruction of 3D image stacks. For multi-modal 3D-reconstruction, the peptide dataset was pair-wise registered with the respective lipid slices of the previously reconstructed 3D lipid image stack. Each of the 10 lipid imzML binary files is about 4.9 GB on disk and each of the 10 peptide imzML binary files is about 2.8 GB. Loading and initialization of a single lipid image into M²aia took 3.26 ± 0.67 seconds for the lipid data and 2.06 ± 0.45 seconds for the peptide data on system B as described in Table 5.1 *Timing experiments on reference dataset*. During the initialization, TIC normalization factors and mean overview spectra are created for each dataset.

Successful image-based registration requires images that are rich and similar in structural features. This can be done in M²aia by fast and interactive exploration of ion-images. For the example data, structure-rich images in the lipid dataset were found at m/z 865$\pm$0.65 Da and for the peptide dataset at m/z 2250$\pm$50 Da. For a rough initial alignment of considerably rotated tissue sections, interactive capabilities to rotate the slices by $\pm$15 degrees around the center were used. Additionally, the non-tissue areas were removed from the ion-image generation process by segmentation of the respective areas, using the segmentation tools provided by M²aia. For mono-modal 3D reconstructions, a reference slice was first selected in the corresponding

M²aia plugin from the list of ordered slices. Starting from this reference slice, adjacent slices were automatically aligned to each other by rigid and deformable image-based registration. The process is applied to the image stack in both downward and upward directions (see 3D reconstruction section of Figure 5.8 *Multi-modal 3D image reconstruction*). Rigid registration is based on a multi-resolution registration strategy (Gaussian pyramid with three levels and downsampling factors of 4,2,1). Advanced Mattes Mutual Information is used as metric for the optimization of a Euler transformation using linear interpolation and 250 iterations[88]. For the subsequent deformable registration steps, the same multi-resolution scheme and metric are applied. As deformable transformation, a recursive B-Spline transformation is used with final grid spacing on the original resolution set to 0.8mm, with scaling factors per pyramid level of 2, 1.5 and 1, respectively. Interpolation is performed by third-order B-Splines. The optimization is run for 750 iterations. Figure 5.8 *Multi-modal 3D image reconstruction* summarizes the workflow. To quantify the accuracy of the registration, an interactive selection of seven reference points were made in each slice and in both modalities independently (a subset of points share a common anatomical location in both modalities), resulting in 70 reference points per set. In Figure 5.9 *Target Registration Error - Reference Points* the reference points of both modalities are shown in context of the reconstructed lipid dataset for the mid-slice of the stack. For the mono-modal 3D reconstructions of the lipid and peptide datasets, a TRE of 28 ± 8 µm and 35 ± 5 µm were obtained, respectively, and for the multi-modal reconstruction a TRE of 39 ± 4 µm. A protocol showing how to perform the interactive steps in M²aia of the workflow as described above is available on protocols.io[161]. In addition to the traditional 2D and 3D perspectives MITK provides, volume rendering for three-dimensional image data is also supported. This feature enhances the visualization of spatial relationships within the data, as demonstrated in Figure 5.10 *Visualization of 3D MSI datasets*.

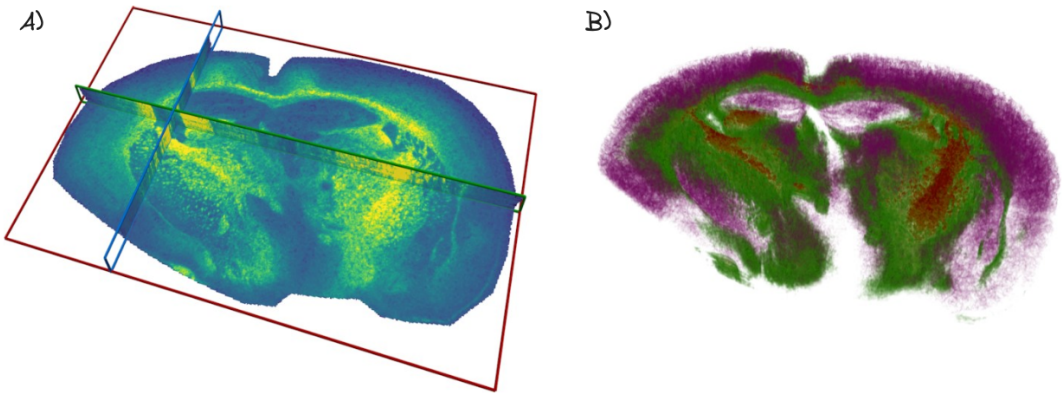

FIGURE 5.10. 3D reconstruction of lipid MALDI-MS TOF images of 10 consecutive brain tissue sections of the *Mouse Brain Dataset*[128] (see subsection 5.1.1 *Lipid/Peptide 3D APP NL-G-F Mouse Brain*) . (A) Multi-planar reconstruction of the 3D MS image showing ion-images at m/z 865.05 ± 0.1 Da. (B) Volume visualization of different mass features. High intensities (green) and low intensities (red) in ion-image at m/z 865.05 ± 0.1 Da are visualized. Additionally, the cortex is highlighted by high intensities at m/z 868.76 ± 0.1 Da (black). Processing and visualization have been performed completely within M²aia. Illustration in Cordes *et al.* (2021)[115] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

5.2.5 Use-case: Semi-automatic multi-modal image registration

To demonstrate the capabilities of M²aia in a multi-modal image fusion use-case, the application of the provided interactive multi-modal capabilities were used to support a co-authored publication published by Abu Sammour *et al.* (2023)[150]. As stated in this work, the data acquisition and processing were in concordance with the declaration of Helsinki and was approved by the Ethics Committee at Heidelberg University, Germany (applications S130/2022 and AFmu-207/2017)[150].

In this use-case, annotations were created by an expert neuropathologist on H&E stained images of human glioblastoma specimens. The research of which this thesis is a part required the transfer of these annotations to the MSI datasets. This was done using the methods available in M²aia, which allowed the integration of the optical images previously used for MSI device acquisition with the H&E stained images. The optical images had an spatial resolution of 5 µm and are intrinsically registered with the MSI data. To provide data within M²aia for the image-fusion, the whole-slide image import utilities (see subsection 3.2.6 *Data import of whole slide images*) are used to convert the RGB channels of the H&E stained and optical images into luminance images. The whole-slide import dialog is shown as an example in Figure 5.11 *Import dialog for whole-slide images*.

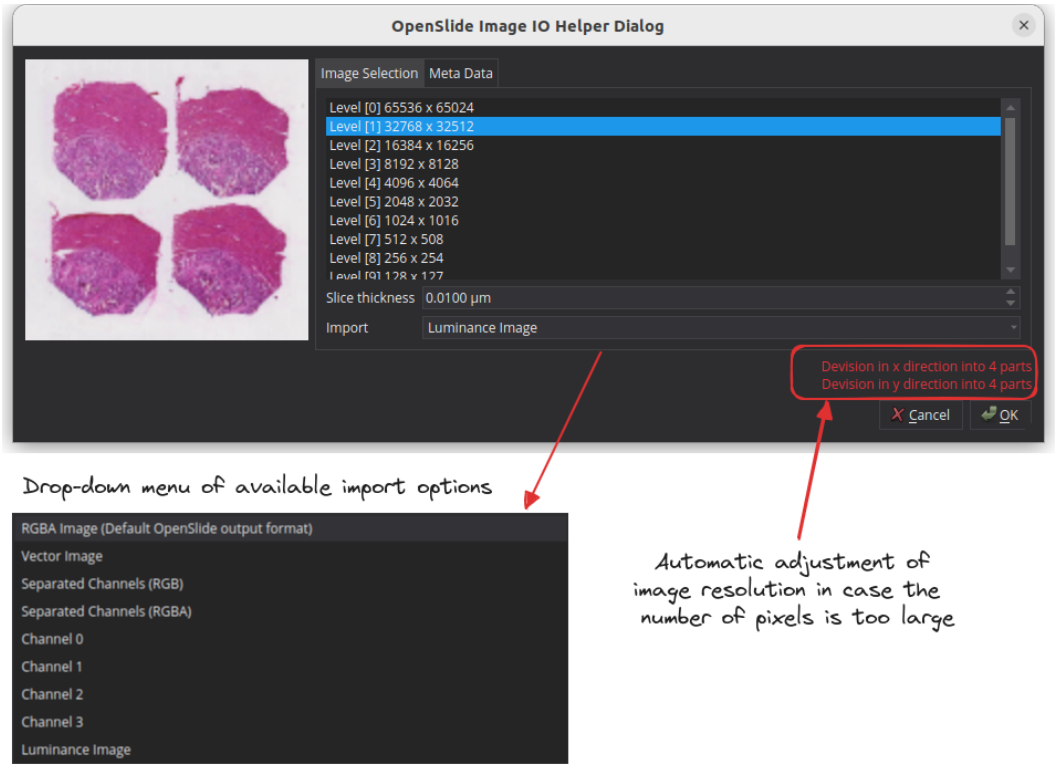

FIGURE 5.11. Import dialog for whole-slide images. The depicted H&E stained images and DESI-MSI datasets were published by Oetjen *et al.* (2015)[33] and is available in the MetaboLights repository [MTBLS282][i]

Before image registration, the images are cropped by using an minimal bounding box around the sample region in order to facilitate the registration process (removing irrelevant parts of the otherwise larger image regions). Furthermore, the resolution of all images were reduced by resampling to a lateral resolution of 7.5 µm. The image-based registration process was then performed as it is described in Abu Sammour *et al.* (2023)[150]: "The full registration is composed of a rigid step, followed by a deformable

step. Each registration results in a set of parameters describing the transformation from the H&E to the MSI image domain. Those parameters are used to transform point information accordingly. Transformed polygons and corresponding annotation labels were written to mis-files. Rigid registration is based on a multiresolution registration strategy (Gaussian pyramid with three levels and down-sampling factors of 4,2,1, each of which represents pattern information at a different scale allowing for a course-to-fine image registration paradigm). The Advanced Mattes Mutual Information in elastix was used as multimodal metric for the optimization of a rigid transformation using linear interpolation and 250 iterations. For the subsequent deformable registration steps, the same multiresolution scheme and metric were applied. For the deformable transformation, a recursive B-Spline transformation was used with interpolation using third-order B-Splines. The optimization was run for 750 iterations."

In some cases of the used dataset, the registration failed due to overlapping tissue regions and displacements. Based on the proposed concepts for registration evaluation and interactive corrections (see subsection 3.3.4 *Evaluation and interactive correction*), the registration process was successfully guided by manually placing pair-wise landmarks in both image modalities.

5.3 Python-based Access to imzML for Deep Learning Applications

Exemplary applications of increasing complexity have been realized to showcase the capabilities of pyM²aia, utilizing openly available MSI datasets published by Geier *et al.* and introduced in subsection 5.1.2 *Adult Earthworm (L. rubellus)*. Each application is realized as an iPython Notebook[145] and are available open accessible on Github[162]. The examples provided demonstrate the practical application of pyM²aia's API and focus on the generation of MSI data samples for deep neural networks based on the strategies described in subsection 3.5.3 *Data Access Strategies for MSI Data*. It is important to note that these examples do not explain or evaluate the methods themselves. Detailed discussions of the methods can be found in the original publications.

A feature comparison is shown that contrasts the capabilities of pyM²aia to pyimzML in section 5.3.1. In the first example (section 5.3.2), the use of pyM²aia to retrieve imzML metadata is demonstrated. The second example (section 5.3.3) demonstrates the application of pyM²aia's signal processing methods. The third example (section 5.3.4) explains the generation of ion-images and demonstrates how to overlay multiple ion-images to show co-localization of ions. Subsequent examples are demonstrated that focus on deep learning applications using pyM²aia. The fourth example (section 5.3.5) showcases a spectral strategy of an autoencoder model for peak learning, the fifth example (section 5.3.6) demonstrates a spatial strategy of an ion-image clustering approach, and in the the sixth and seventh example spatio-spectral strategies are demonstrated for an unsupervised auto-encoder and a supervised model for pixel-wise classification (section 5.3.7).

Availability of supporting source code can be found in section 5.6 *Code and Data Availability*. In all examples the *Earthworm Dataset*[26] (see subsection 5.1.2 *Adult Earthworm (L. rubellus)*) is used.

5.3.1 Feature Comparison: pyM²aia vs. pyimzML

Table 5.3 provides a comparison between pyM²aia and pyimzML, as far as this is possible: pyimzML supports only loading of imzML datasets, whereas pyM²aia additionally supports signal-processing (different methods for baseline correction, normalization, smoothing), the access to helper images (normalization image, index image, mask image) and provides data structures for the spectra, spatial and, spatio-spectral strategies.

5.3.2 Metadata Extraction

The pyM²aia package enables the extraction of crucial image and imaging related metadata from MSI datasets, a feature particularly valuable for researchers analyzing existing data collectives or unseen data in imzML format. The metadata accessible through pyM²aia includes:

- Pixel Spacing (Spot Size): This refers to the distance between the centers of adjacent pixels in an image, which is critical for understanding the resolution and spatial accuracy of the MSI data.
- Image Dimensions: pyM²aia can retrieve the total number of pixels in the spatial dimensions of the MSI dataset, providing insights into the overall size and scale of the sample being analyzed.
- Spectrum Depth: This attribute describes the number of data points in each spectrum, which can indicate the level of detail captured in each pixel's spectral data.
- Spectrum Type: The library supports distinguishing between continuous, processed profile, and centroid spectrum types, as outlined in the imzML standard[53]. Each type represents different methods of spectrum representation and data processing:
  - Continuous Spectrum: A complete representation of the spectral data often without any processing to reduce data size.
  - Processed Profile Spectrum: Spectrum data that has been processed to enhance certain features or reduce noise.
  - Centroid Spectrum: A form of spectrum where data points are condensed into peaks, representing the most significant parts of the spectrum.

Additionally, pyM²aia provides access to image context metadata, which includes all in the data available tags and values as defined in the imzML specification[53]. These tags can contain a wealth of information about the sample, such as the type of tissue, experimental conditions, and specific annotations relevant to the study, enhancing the richness of data available for analysis. The iPython notebook for example I can be found on Github[162].

5.3.3 Signal Processing

pyM²aia provides access to M²aia's optimized signal processing utilities section 3.2 *Concepts for MSI Data Processing*. Example II outlines how to configure the signal processing pipeline. Results of several signal processing configurations are shown in Figure 5.12 *Example II: comparison of mean overview spectra from the same dataset using different signal-processing methods. Mean overview spectra for Section 1 within the range*

Feature	**pyM²aia**	**pyimzML**
Lazy loading (low memory profile)	yes	yes
Spectrum access	yes	yes
Create ion-images	yes	yes
Signal processing	Normalization, Baseline correction, Smoothing, Intensity transformations	no (*)
Overview spectra	Mean, Max	no (*)
Normalization maps	TIC, Sum, Mean, Max, RMS, Internal	no (*)
Spectrum Generators (Spectral and spatio-spectral strategy)	continuous profile and centroid data	no (*)
Ion-image Generators	Spatial strategy, continuous/processed centroid and continuous profile data	no (*)
Average time to load all four datasets (**)	5.2 [seconds]	14.6 [seconds]
Average maximum memory usage (**)	640 [Megabyte]	577 [Megabyte]
ImzML metadata queries	All XML elements with "IMS:.." and "MS:..." tags	Tags related to correctly represent the image (max count of pixels x/y, max dimension x/y, pixel size x/y/z)
Notes	* Not in scope of the package ** by default, M²aia performs a full parse of imzML xml tags, creates an index image, normalization images (for all implemented normalization methods) and overview spectra (max/mean). For comparison with pyimzML, we implemented these functionalities directly in Python with numpy. All four data sets were loaded sequentially. The runtime and maximum memory usage was averaged over 50 repetitions.	

TABLE 5.3. Feature Comparison: pyM²aia vs. pyimzML. System configuration: desktop PC, Ubuntu 22.04, AMD R © Ryzen 9 5900x CPU at 3.7 GHz 12-core processor, 32 GB physical memory, M.2 SSD, and Nvidia Titan RTX. Table in Cordes *et al.* (2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0)

of m/z 200 to m/z 270 are shown for the Earthworm Dataset[26] (see subsection 5.1.2 Adult Earthworm (L. rubellus)) . Illustration in Cordes et al. *(2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).* Currently, different types of baseline-correction, signal smoothing, normalization, pooling and intensity transformations are supported as introduced in subsection 3.2.2 *Signal Processing*. The iPython notebook can be found on Github[162].

5.3.4 Ion image Generation

Ion-image generation and how to combine multiple ion-images into a colored image (see Figure 5.13 *Creating ion-images with pyM²aia: Example III*) is demonstrated in example III. Generated images can be written in common image formats (e.g., Nearly Raw Raster Data [*.nrrd] image file format) that are compatible with M²aia, enabling interactive exploration of image artifacts generated with pyM²aia using the desktop application M²aia. The iPython notebook can be found on Github[162].

5.3.5 Spectral Strategy - Autoencoder for Peak Learning

The fourth example showcases a spectral strategy (see subsection 3.5.3 *Data Access Strategies for MSI Data*) employing pyM²aia to feed spectra to adapted versions of an autoencoder model for peak learning, as proposed by Abdelmoula *et al.* (2021)[63]. The TensorFlow[163] implementation of the original approach, which loads datasets in HDF5[164] format, has been adapted by replacing the HDF5 input with pyM²aia's imzML reader. The training stability on the dataset by Geier *et al.* has been improved by replacing the originally used categorical cross-entropy loss with mean-squared error loss and removing the sigmoid activation function of the output layer. This example also highlights the use of pyM²aia's spectrum batch generators (Figure 5.14 *Spectral Strategy: Example IV - Concept Implementation*) and demonstrates how to train models individually for each imzML image and how pyM²aia facilitates processing multiple images simultaneously to create a single model for all inputs. Results of the peak learning process are shown in Figure 5.15 *Spectral Strategy: Example IV – Peak Learning* and Figure 5.16 *Spectral Strategy: Example IV - Latent Space*. All changes to the original peak learning code-base of Abdelmoula *et al.* (2021)[63], are available in a Github fork[165]. The iPython notebooks can be found on Github[162].

5.3.6 Spatial Strategy - Ion-image-based Co-localization

The fifth example demonstrates the spatial strategy (see subsection 3.5.3 *Data Access Strategies for MSI Data*) by adapting a PyTorch[166] implementation of an ion-image clustering approach proposed by Hu *et al.* (2022)[58]. This approach uses a pre-trained EfficientNet model[167], fine-tuned with contrastive learning (SimCLR[66]), which relies heavily on data augmentations added to pyM²aia's ion-image batch generator. The model's input channels were reduced from three (RGB) to one (gray-scale) to suit the dataset's requirements, and augmentation methods were adapted for single-channel inputs (see Figure 5.17 *Spatial strategy: Example V*). The iPython notebooks can be found on Github[162].

5.3.7 Spatio-spectral Strategies

Spatio-spectral strategies (see subsection 3.5.3 *Data Access Strategies for MSI Data*) are demonstrated in the sixth and seventh examples, where pyM²aia's spectrum batch generator is used to generate spatio-spectral samples by providing additional

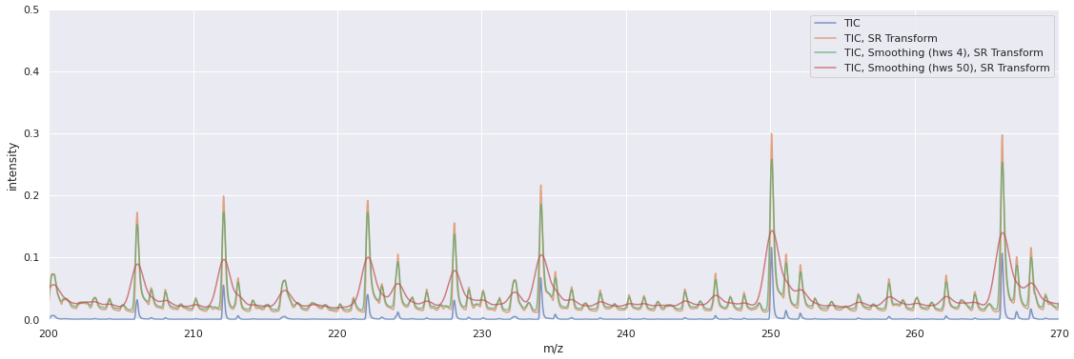

FIGURE 5.12. Example II: comparison of mean overview spectra from the same dataset using different signal-processing methods. Mean overview spectra for Section 1 within the range of m/z 200 to m/z 270 are shown for the *Earthworm Dataset*[26] (see subsection 5.1.2 *Adult Earthworm (L. rubellus)*) . Illustration in Cordes *et al.* (2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0)

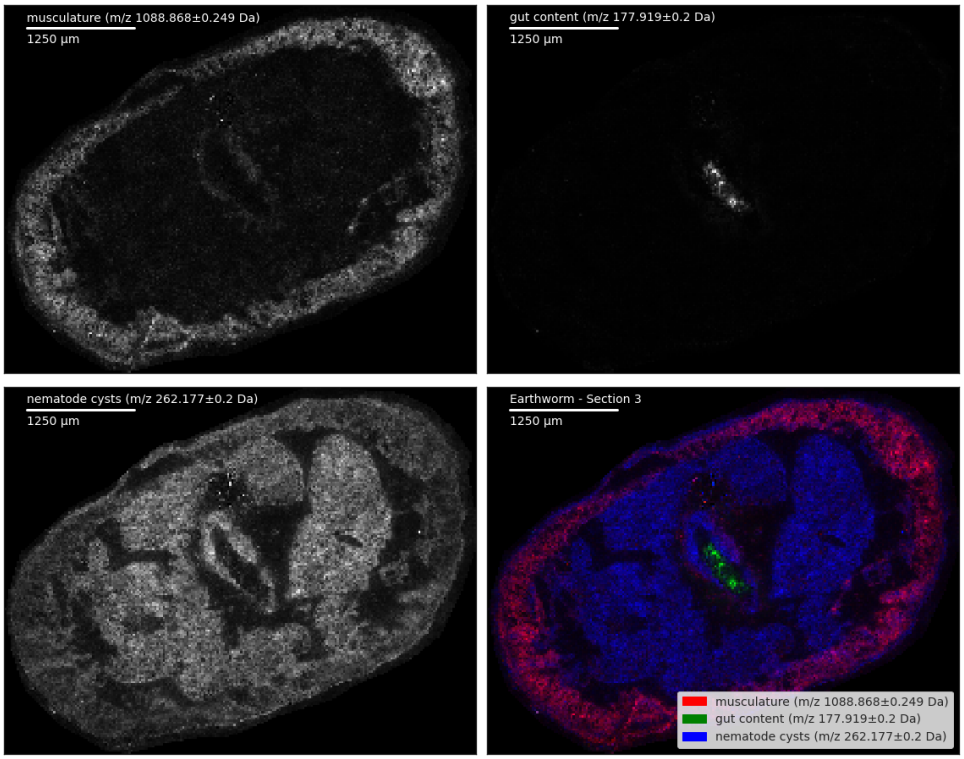

FIGURE 5.13. Example III: sections 1-4 shown for the *Earthworm Dataset*[26] (see subsection 5.1.2 *Adult Earthworm (L. rubellus)*) - combining ion-images for metabolites located within the musculature m/z 1088.868, gut content m/z 177.919, and nematode cysts m/z 262.177 to a single multi-colored representation. Illustration in Cordes *et al.* (2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0)

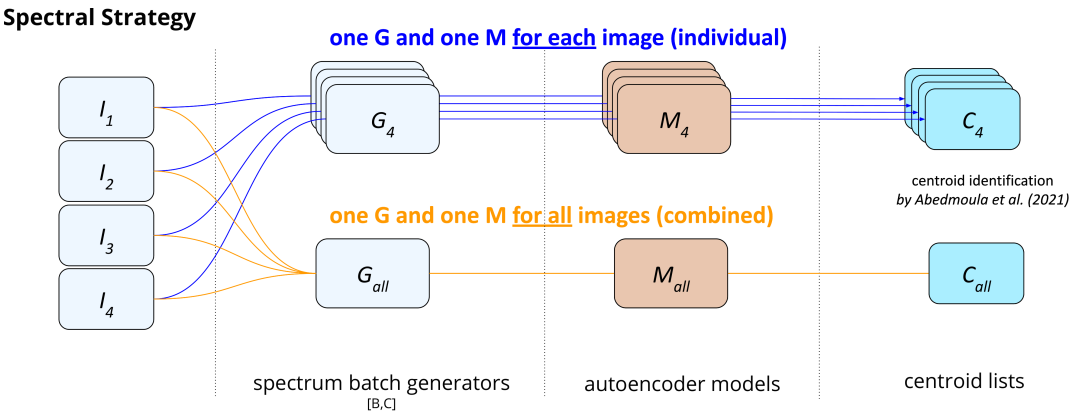

FIGURE 5.14. Spectral Strategy: Example IV – pyM²aia implementations of a spectral strategy for peak learning by Abdelmoula *et al.* (2021)[63]. Target objective is to learn how to reconstruct individual spectra of a set of MSI datasets I_n (four in the example) using an autoencoder model M. The result of the peak learning procedure is a list of centroids. Two different variants are illustrated. The upper blue path shows how to train four independent models M_n using four independent (individual) spectrum batch generators G_n of pyM²aia. The lower orange path uses a single instance G_{all} of a pyM²aia spectrum batch generator to process multiple images at the same time (combined). Illustration in Cordes *et al.* (2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0)

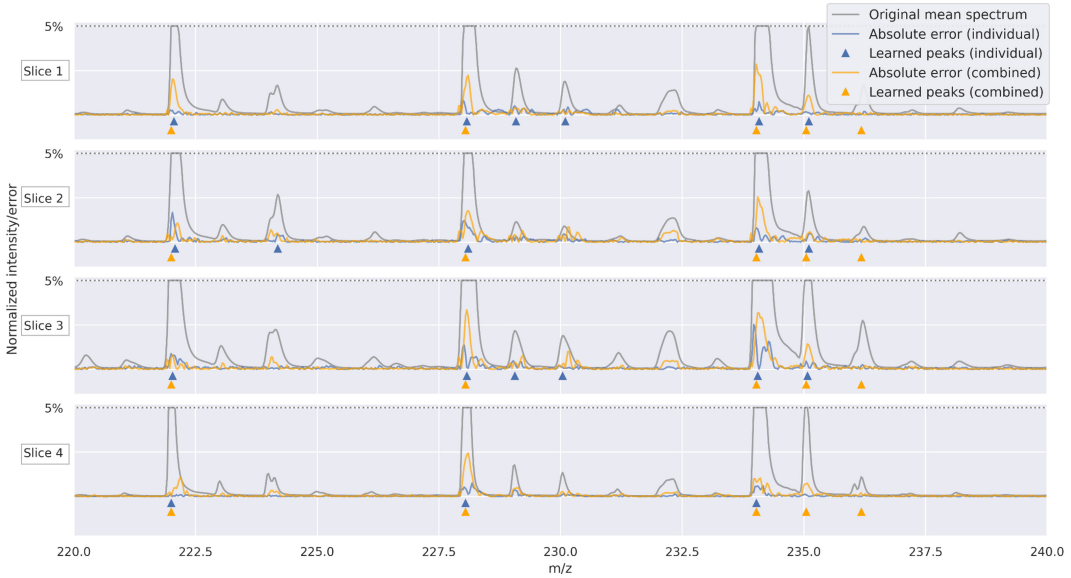

FIGURE 5.15. Spectral strategy: Results of example IV – Peak Learning. Each row represents a single MSI dataset. Absolute errors between reconstructed mean profile spectra (individual models: blue lines; combined model: orange lines) and the original mean profile spectrum (gray lines) are shown for each slice. All values are normalized to 5% of the maximum of the respective original mean spectrum. Mass range *m/z* 220 - *m/z* 240. Learned peaks are shown for individual models (blue markers) and the combined model (orange markers). Illustration in Cordes *et al.* (2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0)

.

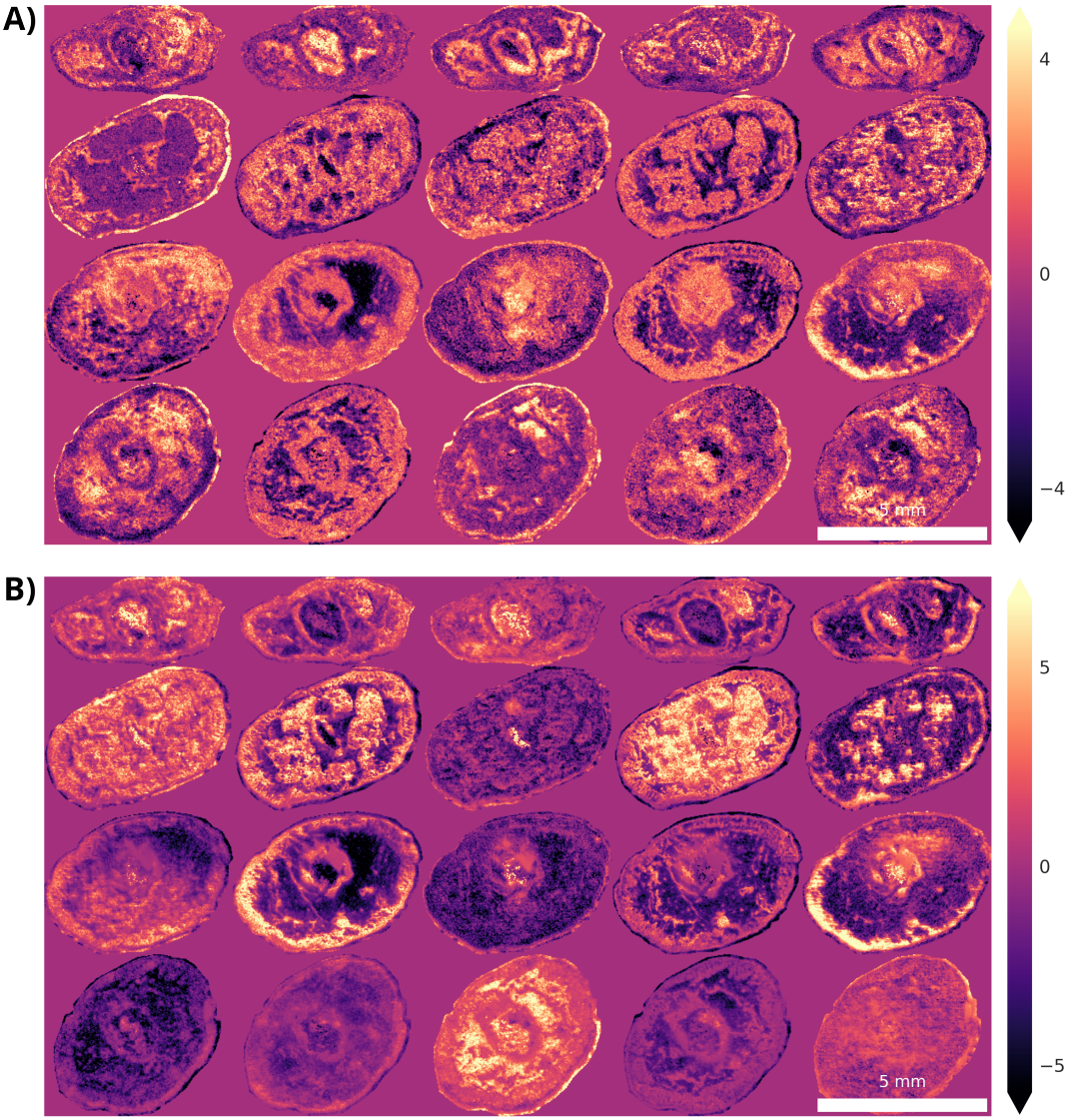

FIGURE 5.16. Spectral strategy: Results of example IV – Peak Learning. Recovered structures of the original high-dimensional MSI dataset visualized with the represented values of the encoded spectra (latent variable z) of variational autoencoders. Each row represents a single slice. Each column represents one component of the encoded latent variable z. From left to right, each row represents the respective component of the latent variable z_0, z_1, z_2, z_3, z_4. In A) individual models and in B) the combined model is used to encode each spectrum of each MSI dataset. Displayed values represent data between the 1st and 99th percentiles. Illustration in Cordes *et al.* (2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

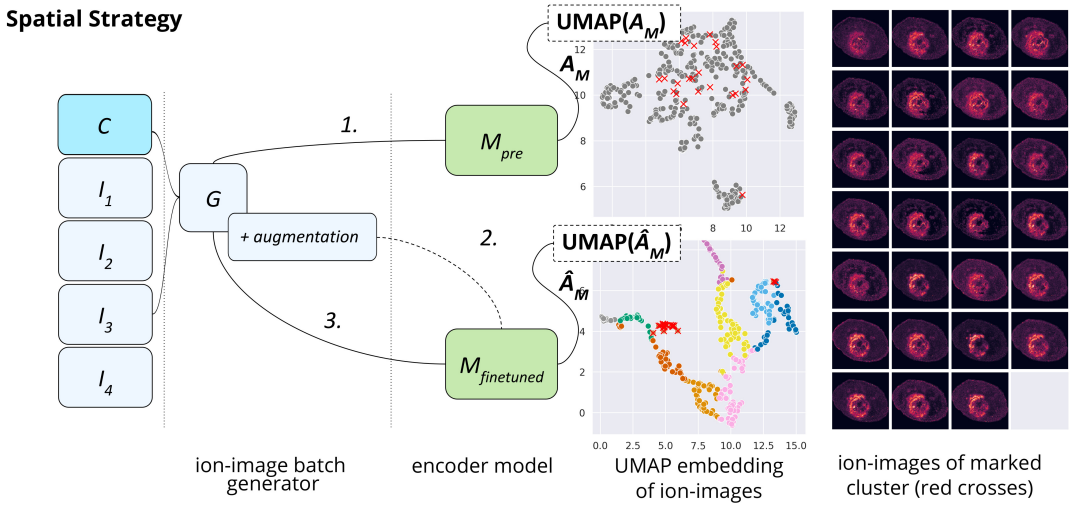

FIGURE 5.17. Spatial strategy: Example V - pyM²aia implementation of a spatial strategy for self-supervised clustering of ion-images by Hu *et al.* (2022)[58]. pyM²aia's ion-image batch generator of MSI dataset I_3 is utilized to feed (1.) a pre-trained EfficientNet M_{pre} model[167] to generate a lower (1024) dimensional embedding A_M of all ion-images generated with respect to a user defined list of centroids C. For (2.) fine-tuning of the model, unsupervised SimCLR[66] training is applied, resulting in (3.) a refined embedding $\hat{A}_M$. Subsequently (fourth column in the figure), UMAP[168] is applied to embed A_M and $\hat{A}_M$ in two dimensions. Each embedded point refers to one ion-image. Spectral clustering[169] was applied to $\hat{A}_M$. The resulting clusters are color-coded in the figure. To visually demonstrate that the clustering in the space $\hat{A}_M$ was successful, the ion-images corresponding to the cluster are marked with red crosses in the UMAP visualisations and are shown in the last column. As expected, the images are visually similar. Without finetuning, these images would not form a cluster, as can be seen by the wide distribution of the marked cluster instances in the UMAP(A_M) visualization. Illustration in Cordes *et al.* (2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

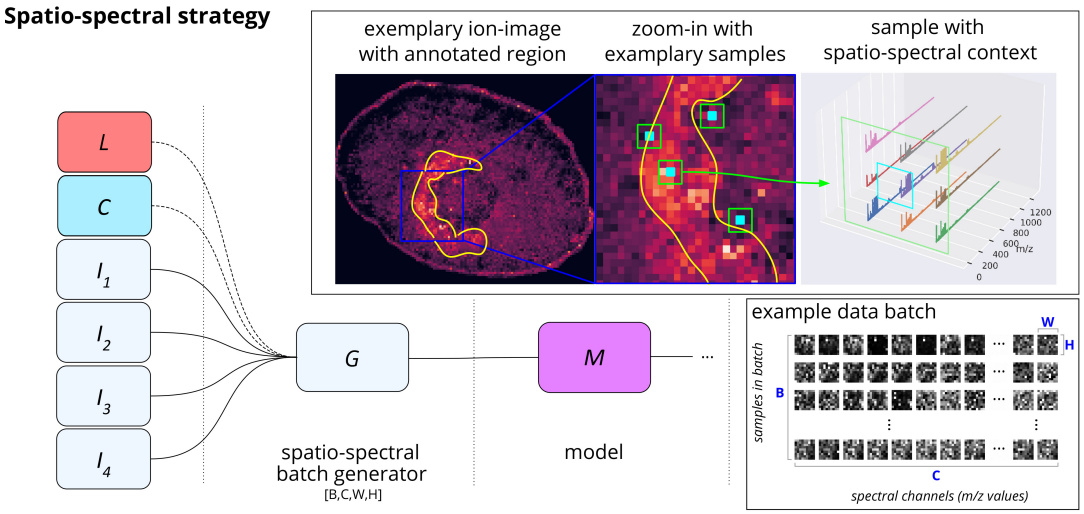

FIGURE 5.18. Spatio-spectral strategy: Example VI/VII - Variational Autoencoder/pixel-wise classification. Generators can be initialized using label images (L) and/or centroid lists (C). For the description of the two examples, see the text. pyM²aia enables individual as well as combined spatio-spectral processing of MSI datasets (as demonstrated for the spectral strategy in Example IV, see Fig. S3). Illustration in Cordes *et al.* (2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

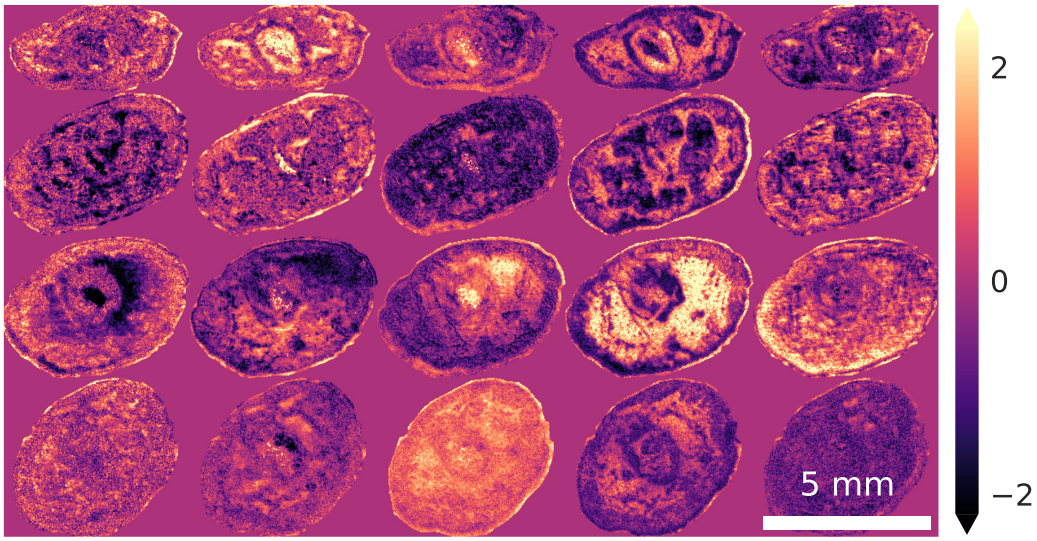

FIGURE 5.19. Spatio-spectral strategy: Results of example VI – Variational Autoencoder. Latent variable z of the variational autoencoder using the spatio-spectral strategy. Each row represents the respective components of the latent variable z_0, z_1, z_2, z_3, z_4 from left to right. Displayed values represent data between the 1st and 99th percentiles. Illustration in Cordes *et al.* (2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

.

neighboring spectra for a given sample location (Figure 5.18 *Spatio-spectral strategy: Example VI/VII*).

Spatio-spectral Strategies - Variational Autoencoder: in example VI, the 3x3 spatial neighborhood of randomly selected spectra are used to train a variational autoencoder. Results of encoded spectra (latent variable z) are shown in Figure 5.19 *Spatio-spectral strategy: Example VI Results*. The iPython notebooks can be found on Github[162].

Spatio-spectral Strategies - Pixel-wise Classification: example VII demonstrates the training of a pixel-wise classification model using spatial annotations on one MSI dataset, applying the trained model on unseen data of all four MSI datasets, and storage of the results including metadata for spatially correct display or further processing. Manual annotations were interactively created for a single sample (slice 3) and exported as a labeled image in NRRD format[i]. Additionally, centroid lists were generated for each sample (slice 1-4) and combined into a single centroid list, exported in text format as comma-separated values (CSV). M²aia was utilized for the interactive creation of labeled images and centroid lists. In the Python notebook of Example VII, all four imzML MSI datasets are loaded, including the labeled image (using SimpleITK[ii]) and the combined centroid list (using NumPy[iii]). The list of centroids and the labeled image are then passed to the spectrum generator, generating batches of the form [X=[B,C,H,W], Y=[B]] (see Figure 5.18 *Spatio-spectral strategy: Example VI/VII*). Here, X represents the spectral data, and Y denotes the labels for each sample in the batch. The convolutional neural network for classification was build using categorical cross entropy, a $9x9$ spatial neighborhood, and randomly selected spectra from the provided annotated regions. Results of the classification approach are shown in Figure 5.20 *Spatio-spectral strategy: Pixel-wise Classification Results of Example VII*. The iPython notebooks can be found on Github[162].

5.4 Integration of Third-Party Image Processing Methods

The main goal of the experiments is to test the applicability of the concepts (see section 3.4 *Concepts for Integration of Third-Party Image Processing Methods*) for the integration of third-party methods. Thus, the perspective of a developer (in subsection 5.4.1)and an user (in subsection 5.4.2) is explored in more detail. The applicability of the concepts is observed from these perspectives and generalized observations were documented, based on the experience of several integrations. Subsequently, exemplary integrations of image processing methods into the UI of M²aia are shown. Publicly available state-of-the-art methods written in different programming languages such as Python and R are encapsulated within Docker-based processing containers and made accessible within the interactive environment of M²aia. The Python-based integration of the UMAP[168] method for dimensionality reduction is demonstrated in subsection 5.4.3. In subsection 5.4.4, the integration of the TotalSegmentator[105] as an MSI-independent deep learning method for segmenting clinical image modalities is demonstrated. MIS-related methods as the *moleculR*[150] framework for creating Molecular Probabilistic Maps (MPMs), are shown in subsection 5.4.5, as well as the deep learning method Peak Learning[63] for MSI datasets (in subsection 5.4.6).

[i] https://teem.sourceforge.net/nrrd/; accessed April 2024
[ii] https://simpleitk.org/; accessed 2024
[iii] https://numpy.org/; accessed April 2024

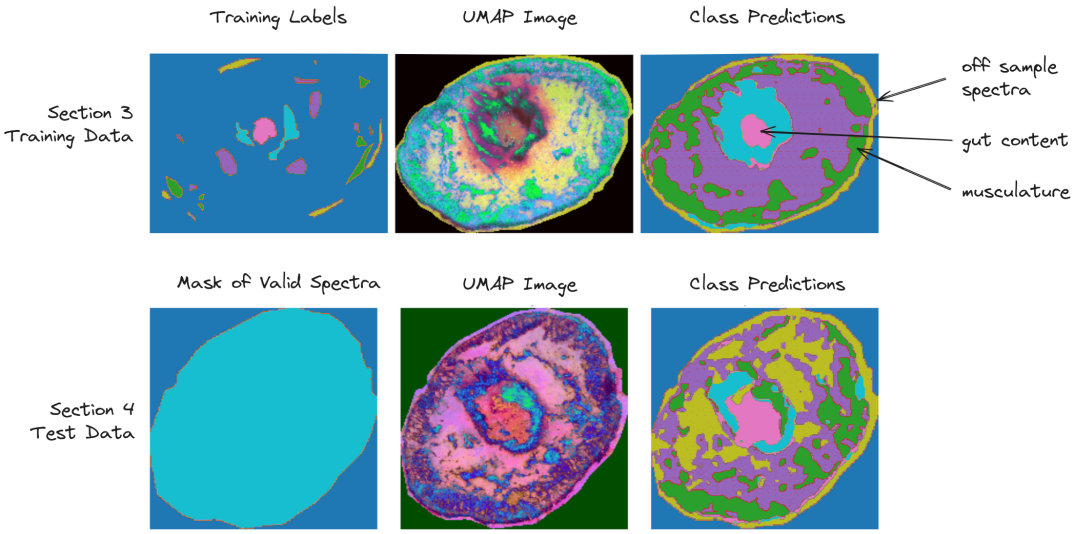

FIGURE 5.20. Spatio-Spectral Strategy: Results of Example VII – Pixel-Wise Classification. Training labels are generated for section 3 of the *Earthworm Dataset*[26] (see subsection 5.1.2 *Adult Earthworm (L. rubellus)*) . For the pyM²aia-based approach, the classification model was successfully applied to section 4 (technical perspective). Illustration created by using figures of the example VII notebook published in Cordes *et al.* (2024)[118] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

.
.

5.4.1 Developer Perspective

From the developer perspective, the integration concept of the new image-processing methods was readily achievable and applicable. The following steps were generally identified: first, the method of interest has to be selected.

Using Third-party Docker Images: in the ideal scenario, a publicly available Docker image already exists (provided by the method creator) and provides a well-defined command-line interface. If the interface accepts data in M²aia-compatible image or image-related data file formats, the development of the GUI integration can be realized directly. This supposes that the Docker image and the image processing method implemented within it have already been designed in such a way that they can be executed via command-line arguments. Furthermore, it is also necessary to pay attention to compatible image formats and, in case of doubt, to make the necessary adjustments in the custom UI on the M²aia side. The utilization of a predefined Docker image is exemplified in the `mitk-docker`[i] application and subsequently delineated in subsection 5.4.4 *TotalSegmentator - Segmentation of Clinical Images*.

Creating Docker Images: in the case that no Docker image exists, a custom Docker image has to be defined. The main objective of the Docker image is to provide a runtime environment for the method of interest, that could be provided by choosing predefined Docker images for a specific runtime from Docker Hub or by defining it yourself. For example, well defined Docker-based runtime environments exists for Python[ii] or R[iii]. These Docker images can be extended with custom instructions

[i] https://github.com/m2aia/mitk-docker; accessed April 2024
[ii] https://hub.docker.com/_/python; accessed April 2024
[iii] https://hub.docker.com/u/rocker; accessed April 2024

tailored to the needs of the third-party method being integrated. A custom Docker image based on `python:3.10-bullseye` for integrating pyM²aia-based applications is shown in code listing 5.1.

```
FROM python:3.10-bullseye

RUN apt-get update
RUN apt-get install -q -y --no-install-recommends \  # install dependencies for pyM2aia
    libglu1-mesa-dev \
    libgomp1 \
    libopenslide-dev

RUN pip install m2aia
COPY app.py /app.py
ENTRYPOINT [ "python", "app.py"]
```

LISTING 5.1. This Dockerfile sets up an environment for running pyM²aia by starting from the python:3.10-bullseye base image. It updates the package lists and installs necessary dependencies (libglu1-mesa-dev, libgomp1, and libopenslide-dev). The Python package `m2aia` is installed via pip. Finally, it copies the app.py script (implements the execution of the method-of-interest) into the container and sets the entry point to execute this script using Python.

Data Conversion Strategies: the necessity for data format conversion may arise depending on the image processing method employed. This may necessitate the implementation of new read and write methods, which could be implemented by M²aia or through conversion strategies within the Docker container. As a guideline, the implementation of new data conversion strategies should be carried out by the Docker-image and use widespread and open data formats on the side of the interactive application, to facilitate the reusability of the created Docker images. As an example, a Docker-based data conversion strategy is carried out for the integration of a dimensionality reduction method and is delineated in subsection 5.4.3 *UMAP - Dimensionality Reduction of MSI Datasets*.

Custom User Interface: integration into the UI of the interactive application is done by creating a custom view. This view implements the execution logic within the interactive application and is responsible for data conversions and parameter parsing to the Docker-based processing container. It is required to parse command-line arguments and to control the execution. The creation of new UI elements is highly dependent on the interactive application. Within M²aia, custom views were created using the Qt UI framework and are integrated via the superbuild mechanism of MITK[i]. An example of a M²aia-based custom view for the integration of a dimensionality reduction method can be found in Figure 5.21 *UMAP - Dimensionality Reduction for MSI Datasets*.

5.4.2 User Perspective

From the user perspective, the deployment and operation of all integrated applications within the M²aia platform demonstrated uniformity and robust functionality. The typical workflow begins with the user launching the M²aia application. Subsequently, data are loaded into the system's interactive environment. Users can then explore and manipulate the datasets according to their specific requirements, such as creating spatial annotations or identifying ion-images and peaks in MSI datasets. Upon completion of data processing, the custom view for the integrated applications can be started, presenting user interface elements for input data selection and method

[i] https://docs.mitk.org/nightly/NewViewPage.html; accessed April 2024

configuration. Finally, the resulting data are not only presented for exploration but are also interactively accessible in conjunction with the original data used at the start of the process.

The aforementioned steps are in more detail explained in the following subsections, with the aid of four illustrative examples.

5.4.3 UMAP - Dimensionality Reduction of MSI Datasets

UMAP, short for Uniform Manifold Approximation and Projection[168], is a dimensionality reduction technique commonly used in various fields of data analysis, including MSI. In the context of MSI, UMAP serves several important purposes[54,58,170,171]:

- Visualization of Complex Data: Mass spectrometry imaging generates vast amounts of data, where each pixel can contain information on thousands of m/z values. UMAP helps in reducing the dimensionality of this data to two or three dimensions, which makes it possible to visualize the spatial distribution of molecular species across the sample in a more interpretable form.
- Highlighting Spatial Patterns: by reducing dimensions while preserving the structure of the data, UMAP can highlight subtle spatial patterns and correlations in the dataset that might not be apparent in the original high-dimensional space. This is crucial for identifying areas with distinct molecular compositions.
- Comparative Analysis: UMAP can be used to compare different regions of a sample or even different samples by visually clustering similar features. This is particularly useful in studies involving disease diagnostics, where affected tissues may show distinct molecular signatures compared to healthy tissues.
- Integration of Data: in studies where multiple imaging modalities are used, UMAP can help integrate and analyze data from different sources or different experimental conditions. This assists in obtaining a holistic view of the sample being studied.

UMAP is an interesting method for processing MSI datasets and the integration of this method provides a first example of the potential of the implemented strategy. The data processing in this context adheres to the conceptual framework proposed in subsection 3.2.5 *Data Compression*.

The Docker image is based on ubuntu and provides a runtime environment for Python[i]. The entrypoint script[ii] is written in Python and uses the UMAP Python package for dimensionality reduction[172]. When carrying out an experiment, parameters and runtime-objects within M²aia are made available to the processing container. These are the reference to the folder containing the MSI dataset, a list of centroids generated by peak picking (local maxima; SNR=5), and the parameters for signal processing (maximum pooling; square root intensity transformation; tolerance of 175ppm) of the MSI data and the parameters for UMAP. The list of centroids is used to generate ion images within the processing container, which can then be used in the UMAP method after they have been converted into a feature matrix. After generation, the resulting image is made available by the processing container on the hard disk in the corresponding writable area (workspace) and automatically loaded into M²aia (if output argument was specified as "autoload"; see subsection 3.4.4 *Process Integration*) after the container is closed. The image is at this time point a vector image with three

[i] https://tinyurl.com/m2aia-dockerfile-umap; accessed April 2024
[ii] https://tinyurl.com/m2aia-docker-umap; accessed April 2024

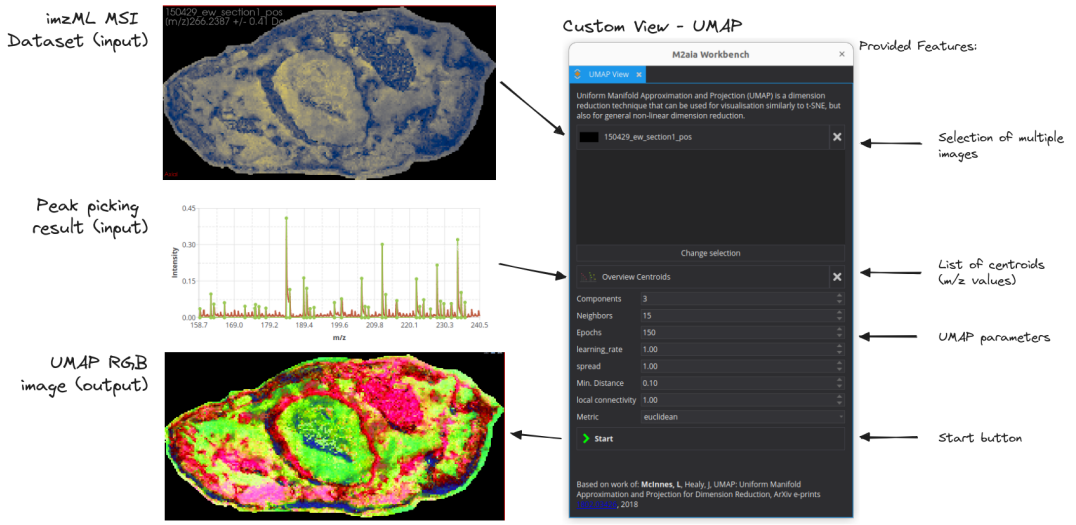

FIGURE 5.21. Dimensionality Reduction for MSI Datasets using UMAP. The custom UMAP view, spectrum view, inputs and output images of the Docker-based integration are shown. The segmentation view shows the mean overview spectrum (red) and peak picking results (green). The bottom row shows the three channels of the dimension reduced image.

components, which is converted to an RGB image and added to the visualization pipeline within M²aia. The custom view for providing controls and parameters and inputs and outputs are illustrated in Figure 5.21 *UMAP - Dimensionality Reduction for MSI Datasets*.

5.4.4 TotalSegmentator - Segmentation of Clinical Images

TotalSegmentator is a deep learning segmentation model for the automatic and robust generation of segmentations of all major anatomical structures in body CT images published by Wasserthal *et al.* (2023)[105]. Since parts of this work were also motivated by a clinical environment (see introduction "Project M2OLIE") and an MSI-independent realization of the integration capabilities was aimed at, a corresponding procedure was provided as a proof-of-concept in this integration task.

In this demonstration the Docker image was already published by the authors of the TotalSegmentator and used directly[173]. The Docker image is provided by the authors of the published method and can be found in the publicly available container registry as announced on the Github project page[174]. The Docker image is based on pyTorch[166] and can be started with the following command (adapted version of the command; on the Github project page[174]):

```
docker run -v /tmp/m2_frLd8h:/m2_frLd8h --gpus device=0 --ipc=host \ # Docker 'run' arguments
    --rm wasserth/totalsegmentator:2.0.0  \                          # Docker image
    TotalSegmentator --ml \                                          # TotalSegmentator arguments
    -i /m2_frLd8h/input_image.nii.gz \
    -o /m2_frLd8h/results.nii
```

LISTING 5.2. Docker run command to start the TotalSegmentator Docker image[173].

When carrying out an experiment, objects from the M²aia runtime environment are made available to the processing container. This is the reference to the folder containing the CT image and the parameters for the TotalSegmentator. The provided Docker image can be configured to use a GPU for model prediction. After generation, the resulting segmentation image is made available by the processing container on the

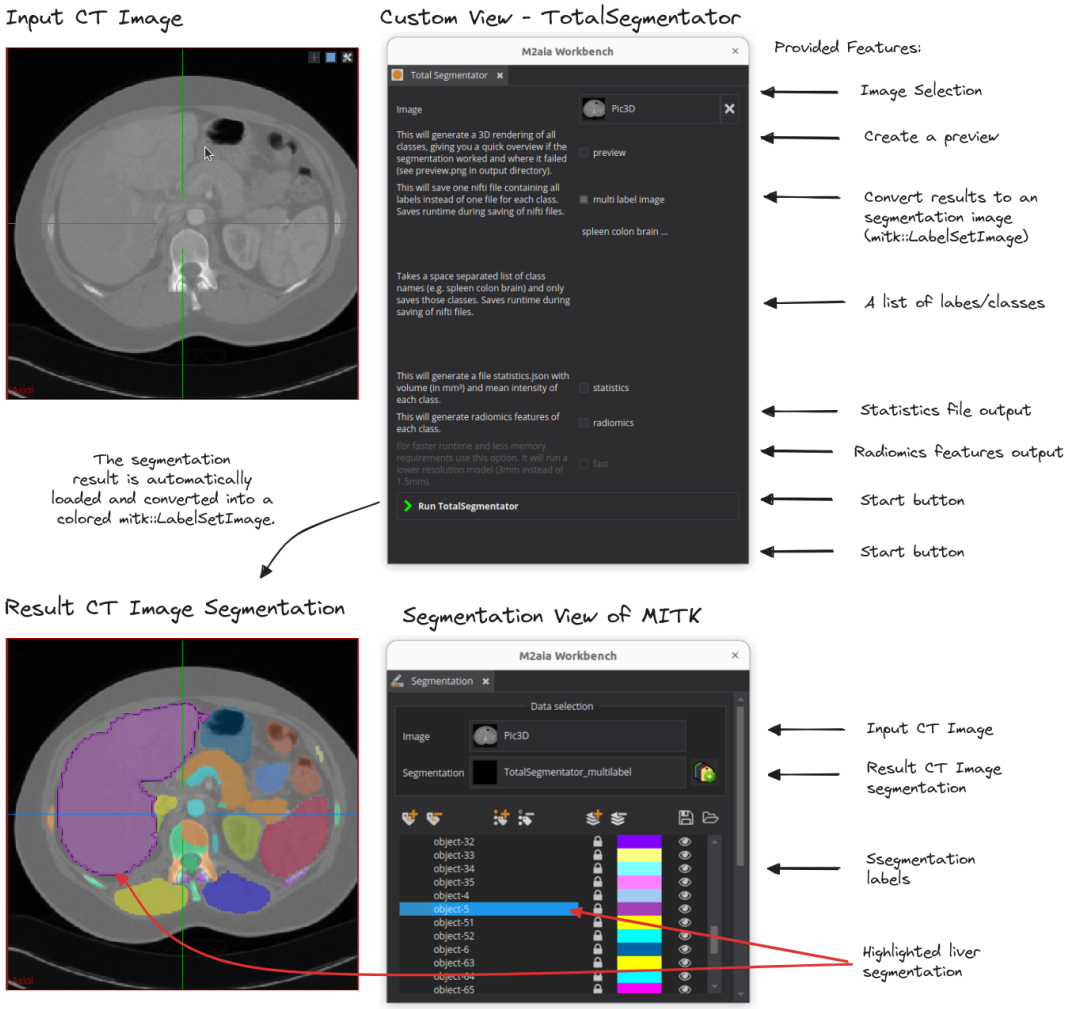

FIGURE 5.22. The TotalSegmentator View, MITK's Segmentation View, and input and output images of the Docker-based integration are shown. The Segmentation View shows the numerical object labels corresponding to the TotalSegmentator output labels. The liver segmentation is highlighted (red arrows).

hard disk in the corresponding writable area (workspace) and automatically loaded into M²aia after the container is closed.

A CT image from the test data project of MITK[i,175] is used to test the integration. The TotalSegmentator view, MITK's segmentation view, and the input and output images are shown in Figure 5.22 *TotalSegmentator - Deep Learning segmentation of a CT image*. The execution code of the integrated TotalSegmentator view can be found in the `mitk-docker` repository[176].

5.4.5 moleculaR - Collective Projections of Metabolites

The moleculaR R package by Abu Sammour *et al.* (2023)[150] provides a computational framework that introduces probabilistic mapping and point-by-point statistical testing of metabolites in tissue via MSI. It enables collective projections of metabolites and consequently spatially-resolved investigation of ion milieus, lipid pathways or user-defined biomolecular ensembles within the same image.

[i]https://phabricator.mitk.org/source/mitkdata/; accessed April 2024

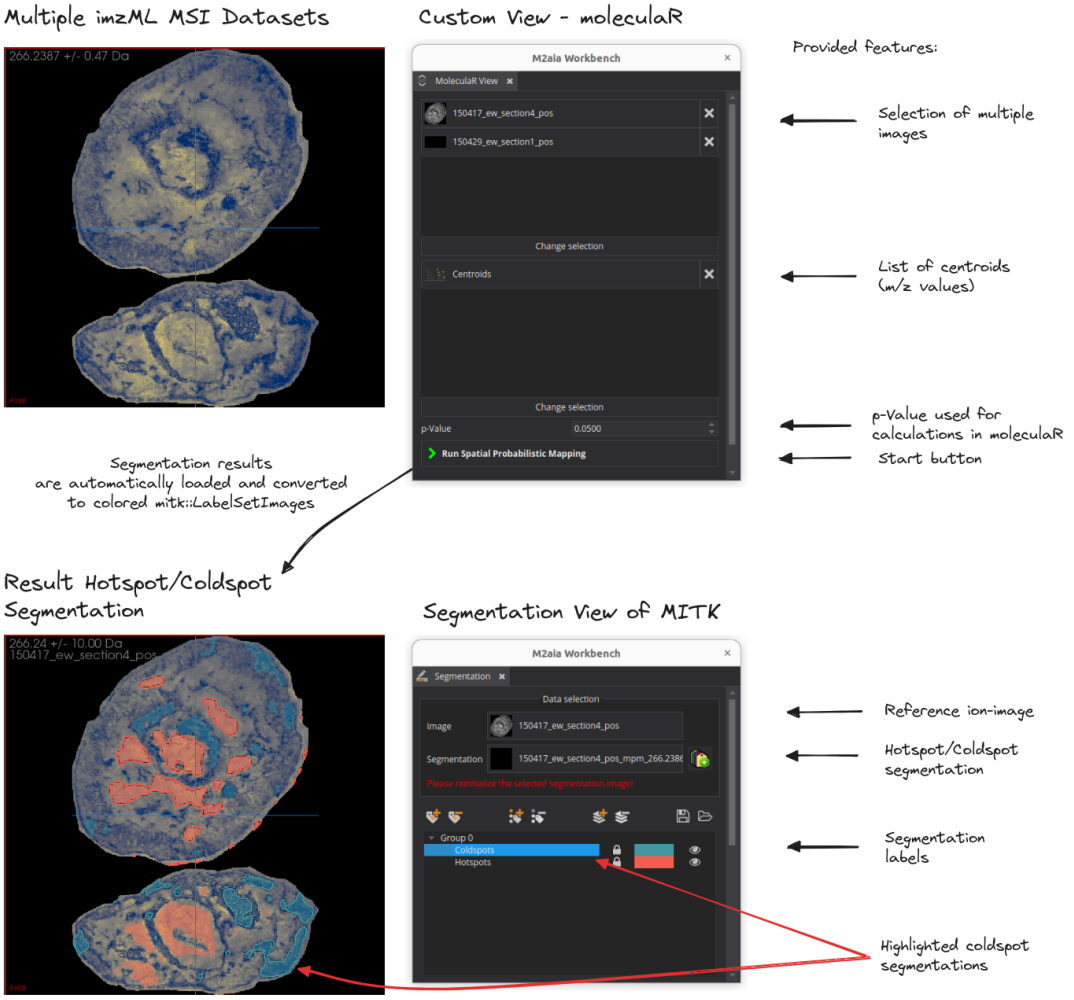

FIGURE 5.23. The moleculaR view, MITK's segmentation view, input and output images of the Docker-based integration are shown. The Segmentation View shows the output hotspot/-coldspot labels of the moleculaR package. The coldspot segmentations are highlighted in blue (red arrows).

In this demonstration, the R package is integrated into a Docker image that supports an R runtime environment[i]. Additionally, an entrypoint script[ii] was composed in R to execute the appropriate method-of-interest. It realizes the command-line argument parsing, image import, data processing, and export of results. Both files are part of M²aia's Github repository[177]. This realizes the objective of programming language independent integration of state-of-the-art methods into M²aia. The custom view of the moleculaR integration, MITK's segmentation view, and the input and output images are shown in Figure 5.23 *moleculaR - Collective Projections of Metabolites*. Results of moleculaR, which include labeled images of hot and cold spots, are incorporated into M²aia, thereby becoming accessible to MITK's segmentation utilities.

5.4.6 Peak Learning - Unsupervised Peak Identification

The peak learning proposed by Abdelmoula *et al.* (2021)[63] was realized as pyM²aia based deep learning application (see subsection 5.3.5 *Spectral Strategy - Autoencoder*

[i]https://tinyurl.com/m2aia-dockerfile-molecular; accessed April 2024
[ii]https://tinyurl.com/m2aia-docker-molecular; accessed April 2024

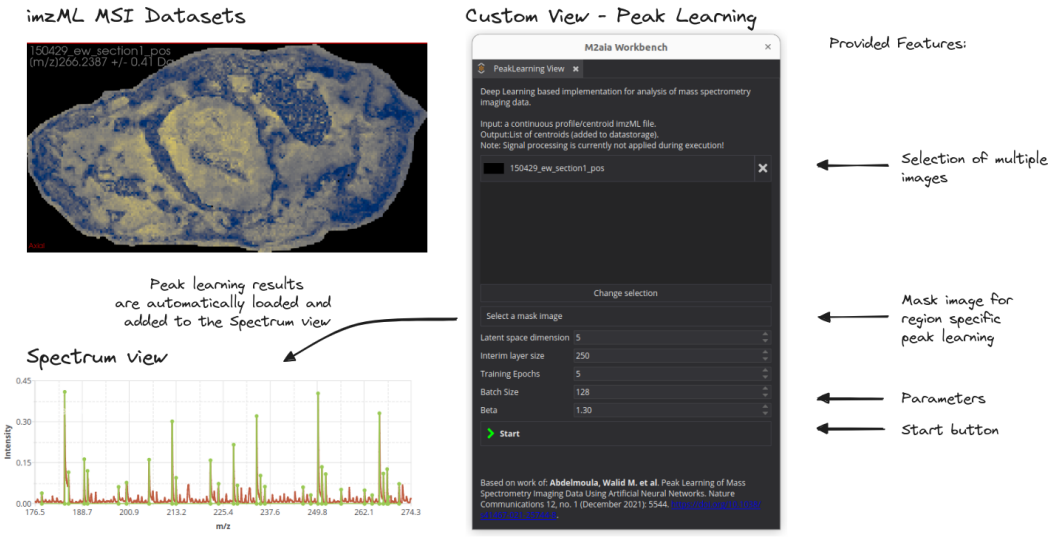

FIGURE 5.24. Peak learning method proposed by Abdelmoula *et al.* (2021)[63] is integrated via the Docker-based system. This figure displays three components: the peak learning view, the spectrum view, and the input image. The spectrum view shows the output mean overview spectrum in red, alongside peaks identified in green.

for Peak Learning) and is integrated in the interactive environment of M²aia, as shown in the previous subsection 5.3.5 *Spectral Strategy - Autoencoder for Peak Learning*. In this example, the integration is based on the `nvidia/cuda:11.8.0-runtime-ubuntu22.04` Docker image in order to support the utilization of GPUs. According to the previous examples, a dedicated UI for parameter and runtime-object parsing was realized. The UI, workflow, and results are illustrated in Figure 5.24 *Integration of the Deep Learning based Peak Learning*. The Docker image and the application code can be found in the Github repository of M²aia[i].

5.5 A Software Ecosystem for Image-based Development, Processing and Collaboration

Thus far, the presented results have demonstrated a variety of concepts for the processing of multi-modal 2D/3D MSI data. Consequently, these implementations provide a feature-rich environment in which advanced MSI questions can be processed on local systems. This section presents the results of complementary concepts for a remote working environment as introduced in chapter 4 *Concepts for Interactive Remote Working*. The objective of these concepts is to provide strategies that facilitate the transition of the previously described MSI data processing concepts to centralized resources. These include, but are not limited to, (i) interactive image analysis (introduced in section 3.3), (ii) the integration of programming language-independent image processing methods (introduced in section 3.4), and (iii) the creation of deep learning models (introduced in section 3.5), (iv) the remote development of image processing method, and (v) collaboration with domain experts. These concepts are demonstrated through a series of use-cases that illustrate the potential applications of the proposed remote working strategies, thus building a software ecosystem for remote analysis of MSI data.

[i]https://tinyurl.com/m2aia-docker-peaklearning; accessed April 2024

The individual components of the software ecosystem are shown in subsection 5.5.1 and the structure of the software ecosystem in 5.5.2. Based on the components of the software ecosystem, a remote developing use-case is outlined in 5.5.4, followed by an collaborative use-case of an multi-rater image annotation workflow in 5.5.4 and a remotely executed 3D reconstruction of MSI data in 5.5.5.

5.5.1 Components of the Software Ecosystem

The components of the remote interactive working environment includes multiple software solutions for the concepts introduced in chapter 4 *Concepts for Interactive Remote Working*. In the following paragraphs, the implementations of the components are described in more detail.

Application Controller: the Application Controller utilities (see section 4.3 *Platform Architecture*) are of central meaning to facilitate the user-interactions with remote resources. In context of the proposed software component VS code as a versatile tool for accessing the centralized resources, an Application Controller extension called vcM²aia was created. This Application Controller realizes the management of interactive applications via convenient UI integrations. The UI components of the VS Code extension are illustrated in Figure 5.25 *Illustration of the vcM²ia Application Controller*.

Remote Interactive Applications The interactive applications (see section 4.4 *Remote Interactive Applications*) are Docker-based virtual desktops that can be rapidly deployed, providing a single interactive image processing desktop application. To illustrate the potential for packaging interactive applications into Docker images with regard to the use-cases of multi-modal biomedical image processing, a set of application containers for interactive image processing has been implemented:

- M²aia[115]: provides, as described in section 3-3.3, fast, memory-efficient and interactive access to 2D/3D MSI and MSI related datasets. It supports interactive tasks like 3D reconstruction and multi-modal image fusion as well as non-interactive tasks including signal processing and dimensionality reduction. Additionally, it provides comprehensive image visualizations in 2D and 3D and access to inherited methods of MITK including interactive spatial segmentation and further processing utilities for 2D/3D clinical imaging data. Parts of this application were published in the GigaScience journal[115].
- QuPath[178]: tailored for digital pathology image analysis, QuPath features an intuitive interface for analyzing whole-slide images, with tools for segmentation, classification, and biomarker quantification. This application is used for annotation/segmentation of 2D optical microscopy imaging data. It serves as a sharable annotation software used for multi-modal and interdisciplinary workflows.
- ilastik[179]: it is a software tool designed for interactive image analysis. It provides a user-friendly interface for users to perform tasks such as image segmentation and classification. One of its key features is its ability to combine machine learning algorithms with user input to achieve more accurate and efficient results. ilastik is commonly used in fields such as biology, neuroscience, and materials science for tasks like cell tracking, object detection, and image segmentation.

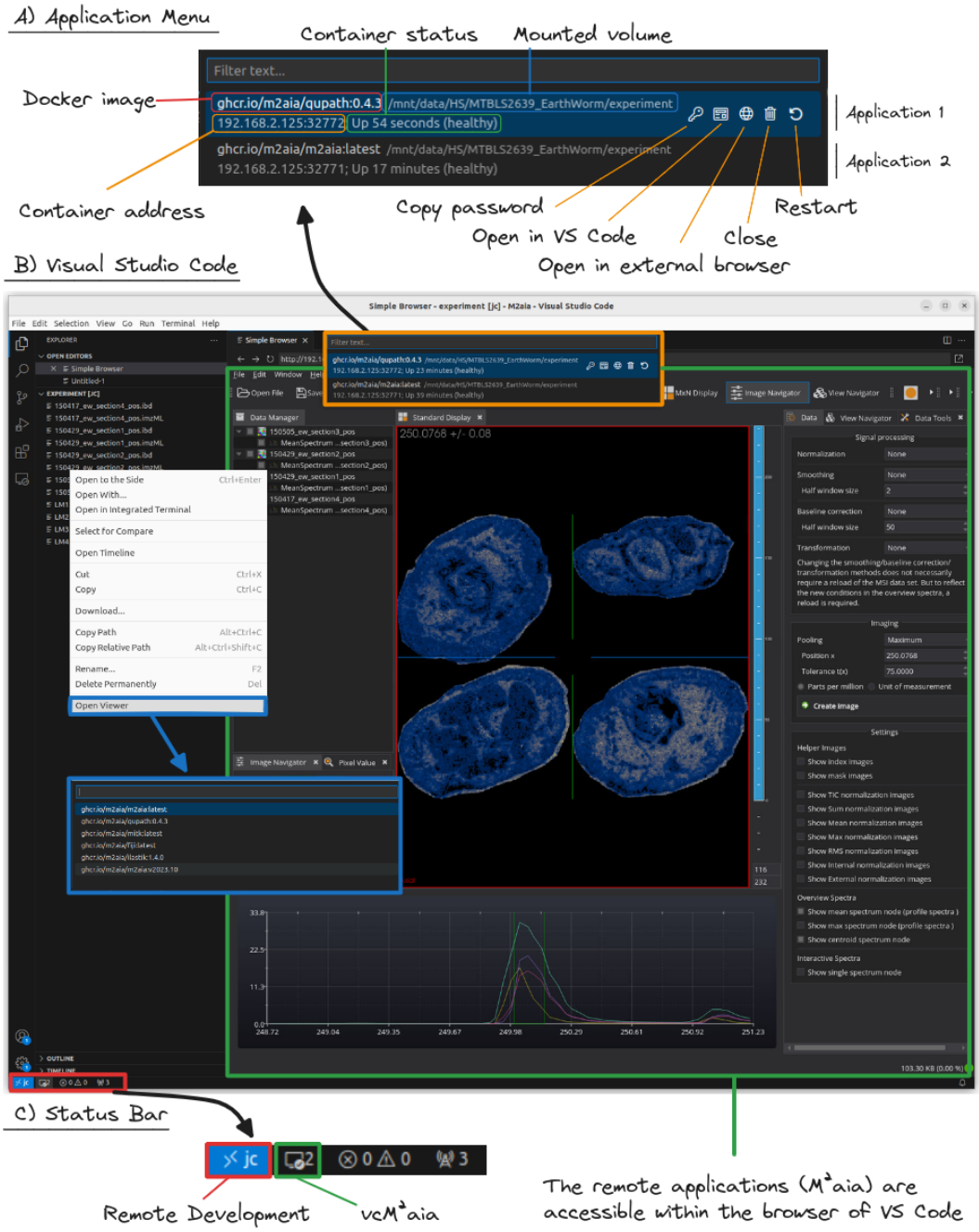

FIGURE 5.25. Illustration of the vcM²ia Application Controller. This illustration of the vcM²ia Application Controller is divided into three sections (A), (B), and (C). VS Code is situated in the center of section (B) and comprises three highlighted areas. The green rectangle represents M²aia, which is currently running on a server and is displayed within VS Code's Simple Browser. A rectangle in the lower left corner in red shows the status symbols of multiple VS Code extensions, which are currently active in the illustrated session. A rectangle in the upper center shows the menu for running interactive applications. Upon right-clicking on a data item or folder within the explorer view of VS Code (accessible via the menu on the left), a context menu will appear. With a blue rectangle highlighted, the quick-access option allows the user to initiate a new remote hosted interactive application by selecting one of the images from the list (blue rectangles). The lower part of this figure (C) shows a detailed view of the VS Code extension status bar. The red rectangle indicates the status of the Remote Development extension of VS Code, which is currently displaying that the active session is remotely accessing a server with the name "jc." The green rectangle indicates the status of vcM²aia, which is currently displaying that two remote applications are running that are associated with the current user. Clicking on the vcM²aia symbol will open the running application menu shown in the upper part of this figure (A).

- ImageJ[89]: an open-source image processing program designed for scientific multidimensional images, ImageJ offers extensive functionality through its modular architecture. It supports a wide range of image formats and provides powerful tools for image manipulation, segmentation, and analysis. ImageJ is widely used in various scientific disciplines, including biology, physics, and materials science, for tasks such as measuring areas, counting particles, and performing complex mathematical operations on image data.

All applications were integrated using the concepts introduced in section 4.4 *Remote Interactive Applications* and are publicly available as part of the Github repository of M²aia[177].

Remote Development Utilities VS Code is a widely-used code editor known for its extensive support for various programming languages through a rich ecosystem of extensions. One of its key features is robust remote development capabilities, which enable seamless coding, debugging, and deployment workflows on remote servers. This functionality is increasingly important for developers who require powerful computing resources or who collaborate within distributed teams. While other solutions, such as JupyterLab and R Studio, also offer remote development capabilities, they lack the proposed Application Controller utilities that are integral for the efficient image-oriented development process.

5.5.2 The Software Ecosystem

The components described in subsection 5.5.1 *Components of the Software Ecosystem* are composed to a software ecosystem. The resulting composition can handle a variety of use-cases. The base structure of this software ecosystem is illustrated in Figure 5.26 *The Software Ecosystem for Remote Analysis of MSI Data*. The illustration shows three distinct section such as (i) the client system, (ii) the remote system, and (iii) the external system of potential Data Users who are located, for example, outside the institute's own network. These illustrative sections are delineated in the following:

- Local System: the (i) local system of a Data Analyst who is acting as the main driver of coordinating individual tasks, requires installations of a VS Code instance and a standard web browser. VS code is used extensive throughout this work and the upcoming use-cases (as mentioned before, other remote IDEs can also be used). VS Code is used as the main interface to centralized resources for remote developing, the remote execution of code and to control the hosting of the interactive applications. The proposed VS code extension vcM²aia (see subsection 5.5.1 *Components of the Software Ecosystem*) offers convenient management capabilities of the interactive application containers. No additionally software components or datasets have to be present on the local system to enable full control of the components of the software ecosystem.

- Remote System: on the (ii) remote system , e.g. a server that holds centralized resources within a research institute, has to provide installations of Docker and the server sided counterpart of the involved remote integrated developing environment. Interactive application images for M²aia, imageJ, ilastik, and QuPath are downloaded to the server on request, that are accessible in the Docker image distribution registry of the Github project[i].

[i] https://github.com/orgs/m2aia/packages

- External System: the (iii) system of an external collaboration partner has to be able to access the network infrastructure of the remote system, e.g. by using a virtual private network connection, in order to access the shared interactive applications. These applications are to be provided by the Data Analyst for a specific dataset using a standard web browser in order to interact with the application containers' HTML5 interface.

The repetitive steps of developing new image processing methods necessitate alternating access to source code, executables, image inputs, and results. The proposed components facilitate the validation of datasets and, based on gained insights, enable the modification of source code and the adjustment of important experimental parameters. This becomes even more important in the event that data size and data access complexity increase, as is the case with hyper-spectral 2D/3D imaging datasets generated by various MSI methods. The proposed software ecosystem provides utilities to rapidly initiate instances of interactive applications on remote-located image data, thus streamlining the aforementioned remotely executed alternating steps. This configuration thereby emulates a local development and image analysis experience. Furthermore, the execution of collaborative tasks with Data Users is supported due to the provided strategies for initializing multiple instances of interactive applications simultaneously on project datasets. To facilitates the monitoring of running interactive applications, the Application Controller (see subsection 5.5.1 *Components of the Software Ecosystem*) offers an intuitive and user-friendly management system.

5.5.3 Use-case: Remote Development of Image Processing Methods

The strategies for the remote development of image processing methods were applied to a range of use-cases delineated in previous sections of the results.

- All examples of creating new deep learning applications using pyM²aia were realized using the software ecosystem (see result section 5.3 *Python-based Access to imzML for Deep Learning Applications*). VS Code was used as a server interface and IDE for creating the comprehensive examples using IPython-based notebooks. The image data were processed and explored using remotely hosted interactive application instances of M²aia, which could be created rapidly on demand. Throughout the development process, the Data Analyst had access to the image data. Centralized GPU resources were utilized during the model training and inference phases.

- The annotation transfer of histological annotations to MSI datasets for the creation of moleculaR[150] (see subsection 5.4.5 *moleculaR - Collective Projections of Metabolites*) was realized remotely in a collaborative setup.

- Furthermore, remote development environments across VS Code, JupyterLab, and R Studio were set up to evaluate their performance and integration capabilities. The following IDE configurations were used:

  - VS Code was configured for remote development using the Remote - SSH extension.
  - JupyterLab[i] and R Studio[ii] were set up for remote development on similar remote servers using Docker images.

[i] https://docs.docker.com/guides/use-case/jupyter/; accessed April 2024
[ii] https://rocker-project.org/images/versioned/rstudio.html; accessed April 2024

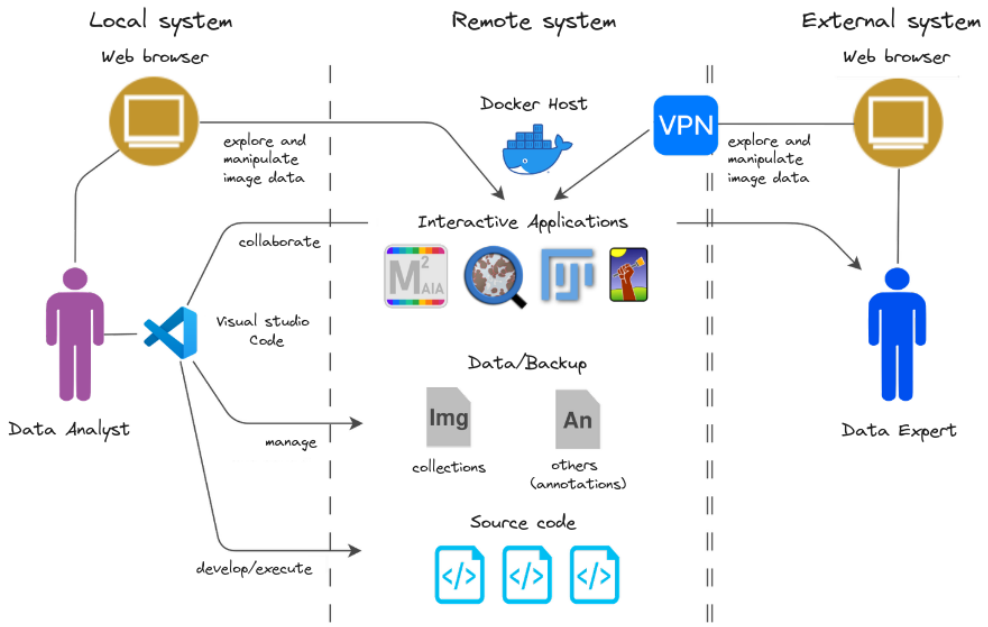

FIGURE 5.26. This diagram provides an overview of the key actions of a Data Analyst that can be realized by using the components of the software ecosystem (see subsection 5.5.1 *Components of the Software Ecosystem*), highlighting the collaborative aspects between Data Analysts and Data Users. The illustration is divided into three areas. The local system of the Data Analyst is on the left, the centralized resources are represented by the institute server in the middle, and on the right, the external system operated by an externally located Data Users. The primary functions of the software ecosystem are illustrated for the roles of Data Analysts and Data Users. As an example, a virtual private network (VPN) connection is depicted to facilitate access to the interactive applications hosted within the institute network. The Data Users can utilize the server address and a password for an application, which can then be opened in a standard web browser. Single dashed line indicates institute-internal access to the remote resources. Double dashed line indicates institute-external access to the remote resources.

The availability and functionality of Application Controller utilities in VS Code were tested. A similar assessment was conducted for JupyterLab and R Studio, but without the convenient access to an Application Controller. The developing of MSI oriented processes showed that VS Code provides seamless support for Python, R, C++, and other programming languages, with straightforward setup and integration for remote development. The VS Code Application Controller utilities, significantly enhance the development workflow by providing rapid access to remote image data. In contrast, JupyterLab and R Studio, while adequately supporting their primary languages (Python for JupyterLab and R for R Studio), exhibited more complex or limited integration processes for the development with support of remote interactive applications, which had to be controlled on the command-line by the Docker API. Additionally, these platforms require a greater initial installation and configuration effort compared to VS Code.

5.5.4 Use-case: Multi-rater Image Annotations

This use-case focuses on the involvement of external collaborators in order to create a multi-rater spatial annotation database of H&E stained histology images. An important aspect of the proposed concept is that desktop applications can be shared online while datasets are never leaving the centralized resources. Since the web browser accessed applications do not offer the download of datasets to the system of end-users, unwanted data access violations are excluded. The download of imaging datasets is only possible by accessing the centralized resources using server credentials, that are trustworthy permitted on individual basis by the administrative personal of the research institution.

Four QuPath application containers were provided, one for each of the four Data Users. Imaging datasets are loaded read-only, while resulting annotations can be written to a dedicated workspace area. After the application container server addresses were shared with the Data Users, user-support was provided utilizing the simultaneous accessibility of application containers for Data Analyst and Data Users. This facilitated the communication process and upcoming questions regarding the annotation instructions, usage of QuPath and targeted annotation quality could be answered rapidly and interactively.

The task was to create multi-rater ground truth for further analysis. The Data Analyst shared the application addresses (e.g. http://<server-ip>:<port>) to the raters, which then can use a standard web browser to access and manipulate a given dataset. No extra software has to be installed for the raters. The interactive applications provide a direct access to the data that is stored on the server and does not need to be fully transferred to a raters system, reducing time, cost and prevents data inconsistencies. Results of the interactive actions are stored on the server for further analysis. Compare Figure 5.26 *The Software Ecosystem for Remote Analysis of MSI Data* and Figure 4.2 *Interactive Remote Working Environment - User Scenario* for this collaborative scenario.

5.5.5 Use-case: 3D Reconstruction of 3D-Cell Cultures

For a study on an integrated platform for 3D reconstructions of 3D-cell cultures by Iakab *et al.* (2022)[180], co-authored by the author, 3D-printed metal casting molds for freezing and embedding (see Figure 5.27 *Sample preparation protocol*) are introduced.

The objective is to assess the platform's capability for preparing spheroid MSI samples with the necessary precision for subsequent 3D reconstruction. Several lipids selectively associated with different cell types or cell-cell interactions within fibroblast and colon cancer biculture spheroids were investigated at a spatial pixel resolution of 20 µm. To achieve this, 100 serial cryosections were prepared and analyzed using MSI. The data was then processed and explored interactively using M²aia to identify structural-rich images suitable for 3D reconstruction (see section 3.3 *Concepts for Image-based Registration in MSI*).

The software ecosystem was prepared for internal collaboration with Data Users. In this case, the Data Users acquired the MALDI-TOF MSI datasets and converted the serial sections into individual imzML files. Data transfer was achieved by uploading files using a web-frontend to a key-value store (database) and was hosted on the institutional server. The Data Analyst provides an instance of the key-value store and one interactive application on the remote system offering access to M²aia via a web browser. The Data Users were able to realize the 3D reconstruction within the interactive framework of M²aia.

Data collected from consecutive sections were normalized by root mean square (RMS) during the import of the individual imzML files (see section 3.2 *Concepts for MSI Data Processing*). The structural-rich ion image at m/z 863.5 was chosen (see subsection 3.2.3 *Ion Image Generation*) for the 3D reconstruction procedure (illustrated in Figure 5.28 *3D reconstruction strategy*). Default parameters were used with a maximum of 20 rigid iterations and 400 deformable iterations. 3D stacks were then exported from M²aia as NRRD[181] files, downloaded to a local system for 3D visualizations as shown in Figure 5.29 *Representation of region-specific ion distributions in space*.

5.6 Code and Data Availability

The contributions of the work are created with the idea of open-source and open-data in mind. This is of central interest in this work and of central importance for the future development of this field of research.

Multi-modal 3D MSI Dataset

- The *Mouse Brain Dataset*[128] (see subsection 5.1.1 *Lipid/Peptide 3D APP NL-G-F Mouse Brain*) is available for download at http://gigadb.org/dataset/100909

M²aia: Source Code Repositories and Desktop Application

- Project home page: https://m2aia.github.io/m2aia
- Biotools (biotools:m2aia): https://bio.tools/m2aia
- The M²aia application (biotools:m2aia) is available at https://m2aia.de
- The M²aia desktop application is an open-source software project in the sense of a 3-clause BSD license. Installer for te application (windows/linux) can be found on Github https://github.com/m2aia/m2aia
- The image-based registration utilities are part of the mitk-elastix extension, an open-source software project in the sense of a 3-clause BSD license. It can be found on Github https://github.com/m2aia/mitk-elastix

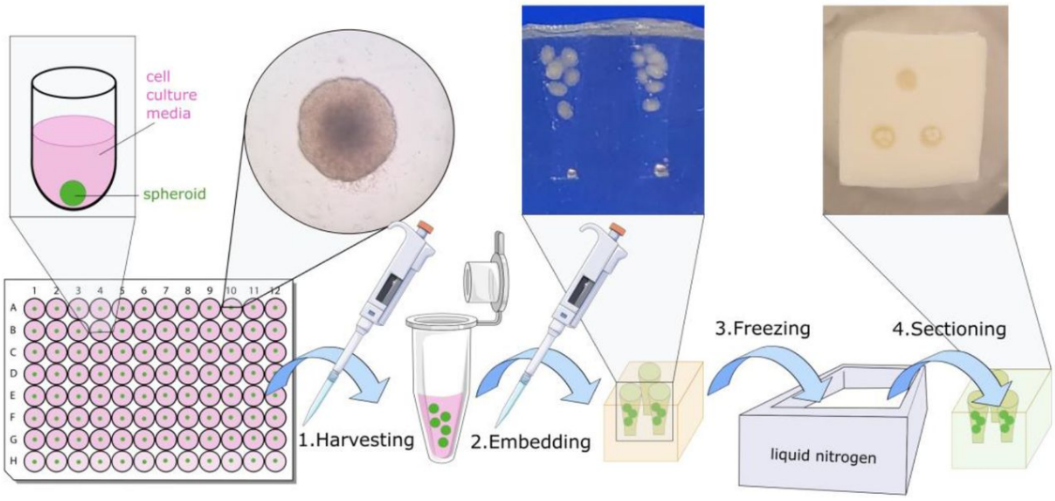

FIGURE 5.27. Sample preparation protocol. Spheroids grown in a 96-well plate are harvested, then embedded in HPMC-PVP-filled gelatin cryo-molds, snap frozen in liquid nitrogen, and finally cryo-sectioned. Inserts showcase pictures from the CeMOS laboratory. Illustration created by Iakab *et al.* (2022)[180] under the conditions of the Creative Commons Attribution (CC BY-NC-ND) license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

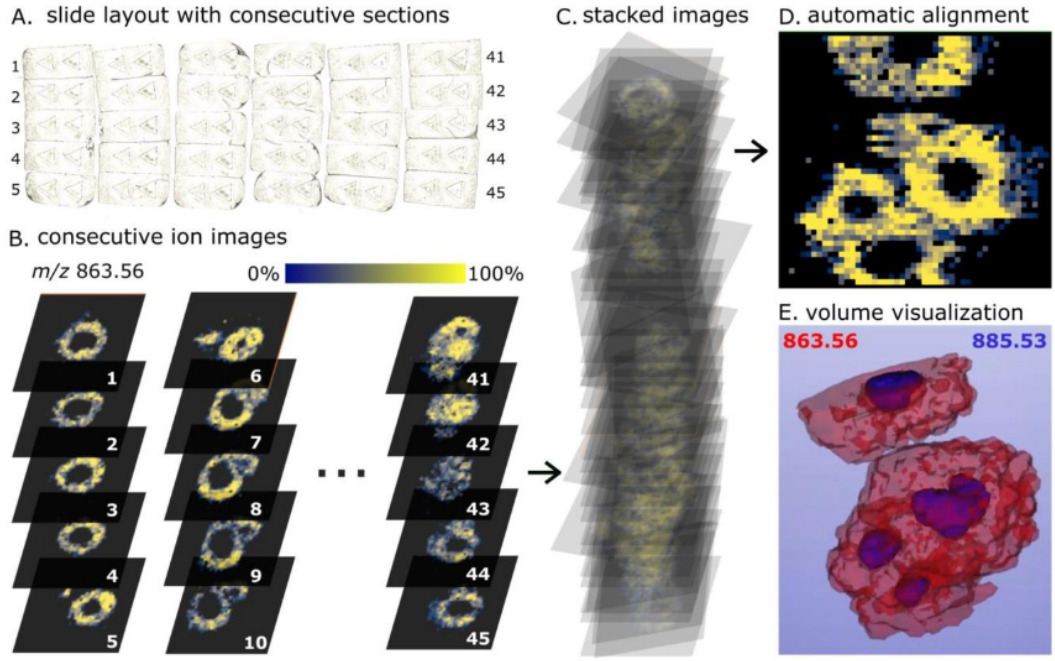

FIGURE 5.28. 3D reconstruction strategy. Slide layout with consecutive sections (A); Consecutive sections illustrated by the ion images of m/z 863.56 (B); automatic stacking and registration of consecutive ion images using M²aia software: illustrative stacking (C) and cross-sectional view of stack (D); volume visualization of different ions of interest, in red m/z 863.56 and in blue m/z 885.53 (E). Illustration created by Iakab *et al.* (2022)[180] under the conditions of the Creative Commons Attribution (CC BY-NC-ND) license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

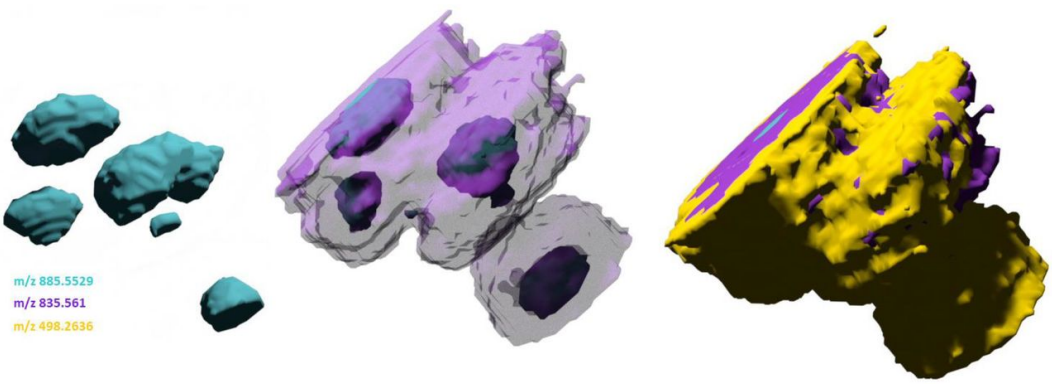

FIGURE 5.29. Representation of region-specific ion distributions in space. The illustration shows frames from a 3D reconstruction video[182] with m/z 498.26, m/z 835.56, and m/z 885.55 describing the exterior layer (yellow), the cancer cells (black), and the fibroblasts (cyan). Illustration created by Iakab *et al.* (2022)[180] under the conditions of the Creative Commons Attribution (CC BY-NC-ND) license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

- The third-party method integration utilities are part of the mitk-docker extension, an open-source software project in the sense of a 3-clause BSD license. It can be found on Github https://github.com/m2aia/mitk-docker

pyM²aia: Source Code Repositories and Package

- The pyM²aia package was published in the Python Package Index (PyPI) repository for (windows/linux) on https://pypi.org/project/m2aia/ (accessed April 2024).
-
- Biotools (biotools:pym2aia): https://bio.tools/pym2aia
- The source code for pyM²aia can be found on Github https://github.com/m2aia/pyM2aia
- The pyM²aia based examples can be found on Github https://github.com/m2aia/pym2aia-examples
- msiPL - Github fork: https://github.com/m2aia/pym2aia-examples-msiPL
- MSI-self-supervised-clustering - Github fork: https://github.com/m2aia/pym2aia-examples-MSI-self-supervised-clustering

Docker Images for Remote Interactive Applications:

- The Docker Image for M²aia is part of M²aia's Github: https://ghcr.io/m2aia/m2aia
- The Docker Image for MITK is part of M²aia's Github: https://ghcr.io/m2aia/mitk
- The Docker Image for QuPath is part of M²aia's Github: https://ghcr.io/m2aia/qupath
- The Docker Image for ilastik is part of M²aia's Github: https://ghcr.io/m2aia/ilastik
- The Docker Image for fiji is just imageJ (fiji) is part of M²aia's Github: https://ghcr.io/m2aia/fiji

All of these guarantees availability, comparability and usability of the developed methods and datasets. All listed web addresses accessed in April 2024.

Chapter 6

Discussion

In this chapter, the findings presented in the previous chapter will be reviewed and their implications for the field of biomedical image processing discussed. The chapter is divided into three parts. The first section of this chapter provides an overview of the results presented in the previous chapter and considers them in the context of the overall concept of the thesis. The second section provides a discussion of the concepts for interactive processing of multi-modal 2D/3D MSI dataset with M²aia, the Python-based access to MSI datasets with pyM²aia and its implications for the filed of deep learning in MSI, and the integration of programming language independent image processing methods into an interactive application context. The third section includes a review of the developed software ecosystem for remote working with biomedical images and its potential applications are reviewed.

6.1 Overview

The primary objective of the present thesis was the development of a software ecosystem for supporting remote executed interactive processes related to multi-modal 2D/3D MSI. All developed software solutions were created with the challenges related to deep learning applications in MSI in mind.

To achieve this objective, an interactive analysis and processing framework for multi-modal 2D/3D MSI was designed. Consequently, the Medical Imaging and Interaction Toolkit (MITK) was extended by the incorporation of MSI data import and processing functionalities, thereby creating the Mass Spectrometry Imaging Applications for Interactive Analysis in MITK (M²aia). The processing of MSI data was enabled within M²aia by implementing an efficient access to MSI data in imzML format (see subsection 5.2.1 *Data Import and Performance*) and commonly used MSI data processing capabilities (see subsection 5.2.2 *Individual Features for MSI Data Exploration and Processing*), including signal processing, visualizations, and analysis steps required in MSI experiments.

The capabilities of M²aia in handling a variety of data processing tasks related to MSI were demonstrated. A biomarker identification use case (see subsection 5.2.3 *Use-case: Biomarker Identification*) was implemented to showcase M²aia's ability to execute a common MSI-related task. This thesis placed a particular emphasis on the implementation of advanced interactive concepts for image-based registration tasks. These included concepts for 3D image reconstructions and visualizations (see subsection 5.2.4 *Use-case: Multi-modal 3D Image Reconstruction*) and concepts for multi-modal image fusion, with the aim of facilitating correlated multi-modal image analysis and annotation transfers (see subsection 5.2.5 *Use-case: Semi-automatic multi-modal image registration*).

A flexible, user- and developer-friendly concept for the programming-language independent integrations of third-party image processing methods was implemented.

This enables the customization of processing workflows by integrating new image processing capabilities within the interactive environment of M²aia (see section 5.4 *Integration of Third-Party Image Processing Methods*).

MSI dedicated deep learning utilities were implemented in Python, which are based on the previously introduced memory-efficient access to MSI datasets in imzML format. Python was chosen to support the widely used Python-based frameworks for deep learning, including PyTorch and TensorFlow. Therefore, the shared library of M²aia was incorporated into a Python context and extended to support the different access strategies, namely the spectral strategy, spatial strategy, and spatio-spectral strategy. These strategies were demonstrated by extensive examples in section 5.3 *Python-based Access to imzML for Deep Learning Applications* on basis of supervised and unsupervised deep learning models.

To enable the transfer of MSI analysis, processing, and development tasks to remote/centralized resources, concepts were implemented that enable interactive remote access to image data, not only for MSI but also for MSI-related imaging modalities. These capabilities were implemented by different software components, collectively forming a software ecosystem for remote working with MSI datasets (see section 5.5 *A Software Ecosystem for Image-based Development, Processing and Collaboration*). To demonstrate the remote working capabilities, three MSI-related use cases were presented, including remotely executed use cases for collaborative tasks, 3D image reconstruction, and development.

In the following sections specifically important aspects are discussed in more detail.

6.2 Interactive 2D/3D MSI Data Processing Application

Interactive access to image data is a key concept in image analysis and provides users with tools for exploration and processing. The objective of this thesis was to create an interactive platform that would facilitate the exploration of multi-modal and 3D MSI-related problems. This would enable users to gain a comprehensive understanding of the data and to facilitate the selection of strategies for solutions on an individual basis.

The proposed concepts make it possible to realize complex/advanced interactive workflows for multi-modal and 3D MSI data visualization and processing, which are dependent on the efficient implementation for imzML data access. This was enabled building upon the advanced software-solutions offered by MITK[92,93]. Although MITK was developed specifically for the processing of clinical images, the potential of MITK is widely utilized and extended by the proposed interactive concepts for MSI-related questions. In contrast to previous approaches, that realize open-source desktop applications for interactive access to MSI datasets[78], the unique characteristics are the strong focus on multi-modality (MSI-to-MSI or MSI-to-Histology) and the interactive concepts for supporting MSI data exploration, analysis, multi-modal image fusion, and 3D image reconstruction.

6.2.1 Interactive Features of M²aia

Parts of the following discussion points were published in Cordes *et al.* (2022)[116] under the conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

M²aia's multi-threading and lazy-loading concepts enable memory-efficient exploration of datasets that are far larger than the system's actual working memory.

As shown in Table 5.3 *Feature Comparison: pyM²aia vs. pyimzML*, loading a 44.2-GB dataset requires <500 MB of RAM. This allows even complex MSI analysis tasks to be performed on local systems. This was demonstrated on the mobile system for m/z candidate detection on an N-linked glycan MALDI-TOF dataset (see subsection 5.2.3 *Use-case: Biomarker Identification*) and 3D multi-modal registration of a lipid and peptide dataset (see subsection 5.2.4 *Use-case: Multi-modal 3D Image Reconstruction*).

Image-to-image registration requires structure-rich images that include common characteristic features between tissue slices or, for the multi-modal case, between modalities. This may require a user-driven search for structure-rich images across the m/z dimension, which is facilitated by M²aia's fast and interactive ion-image generation. Because different masses are intrinsically registered, it is irrelevant whether the structures visible in an image are meaningful entities or imaging/normalization artifacts. To demonstrate this, an unusually wide mass range of 50 Da at m/z 2,250 was chosen to generate a structure-rich image. This was successfully used in subsection 5.2.4 *Use-case: Multi-modal 3D Image Reconstruction* for the 3D reconstruction. Disabling TIC normalization for the same m/z-range leads to a noisy image without structures, not usable for image-based registration—suggesting that the contrast is actually caused by a TIC normalization artefact. Other challenges for purely automatic image registration approaches are significantly misaligned, especially heavily rotated, or, even worse, flipped images. The interactive environment of M²aia makes it possible to quickly obtain a rough pre-alignment of the images that is sufficient as initialization for subsequent automatic refinement. With the possibility to edit the elastix parameter file, M²aia offers unrestricted access to the full potential of the elastix toolkit to enable problem-specific customization of image registration.

Evaluation of registration results is yet another task that requires interaction. It is performed either qualitatively by visualization methods (like blending or checkerboard visualization) or quantitatively by comparing corresponding landmarks or, less accurately, segmentations. Both methods typically require interactive tools, e.g., to select the appropriate parameters for visualization, to define corresponding landmarks, or to perform (or at least verify) segmentations. MITK, the toolkit that M²aia is based on, offers such tools. The applicability of these tools to hyperspectral MSI data, enabled by the data access concepts of M²aia, was demonstrated in subsection 5.2.2 *Individual Features for MSI Data Exploration and Processing* (Image-based Registration Evaluation).

With rare exceptions, transforming an image to another coordinate system requires interpolation of image data. If interpolation is applied to spectral data, the interpolated spectra must be interpreted with caution. To avoid possible misinterpretation of interpolated spectra, M²aia currently calculates only interpolated ion-images and allows the transformation parameters to be stored for use together with the unmodified MSI data. To avoid interpolation of spectra in a multi-modal registration task with MSI and non-MSI data, the MSI image domain should be used as the fixed image domain.

Multi-modal MS imaging refers to approaches with different MSI contrasts (like lipid and peptide MS imaging), as well as combined MSI and non-MS imaging methods, e.g., MSI combined with microscopy. M²aia's capabilities for the former scenario were demonstrated in subsection 5.2.4 *Use-case: Multi-modal 3D Image Reconstruction*. Combining MSI and microscopy is a common multi-modal MSI experiment with its own challenges in interactive visualization. Owing to the high lateral resolution of microscopy images, memory-efficient handling of microscopy datasets requires pyramidal and tiled storage approaches. To enable this in M²aia an interface for reading

whole-slide images by utilizing the OpenSlide library[137] was implemented as demonstrated in subsection 5.2.5 *Use-case: Semi-automatic multi-modal image registration*.

6.2.2 Integration of Third-party Image Processing Methods

In cases where novel image processing methods have been developed in isolation from an interactive context, it can be of great benefit to the users and developers of these processing methods to be able to integrate them into an interactive context by using the proposed interfaces. The interfaces offer a way of integration that is independent of the programming language of the method of interest. The proposed command-line-based implementation of image processing methods in Docker images facilitates the reusability of these Docker-based applications. If the resulting Docker images are publicly available, they can be used in a standalone scenario via the command-line or integrated into applications that follow a similar strategy to the proposed one.

These capabilities has been demonstrated in several use cases that show the integrability of such applications, not only for MSI data processing, as shown for UMAP (see subsection 5.4.3 *UMAP - Dimensionality Reduction of MSI Datasets*) and MoleculaR (see subsection 5.4.5 *moleculaR - Collective Projections of Metabolites*), but also for deep learning applications (see subsection 5.4.6 *Peak Learning - Unsupervised Peak Identification*) and clinical imaging (see subsection 5.4.4 *TotalSegmentator - Segmentation of Clinical Images*). Furthermore, the use-cases demonstrate the generic ability to include processing methods independent of the programming language in which they are implemented. The implementation of the concepts in M²aia was straightforward, thanks to the generic concepts of MITK for reading and writing data in provided image formats. If similar concepts, as in MITK, are available in third-party applications, the integration and utilization of the implemented and future Docker-based image processing methods can be similarly seamless and efficient.

The integration concept have been implemented as a standalone project called `mitk-docker`[117], and was used for the integration of MSI-based processing methods, among others. Previous work, such as that proposed by Razeghi *et al.* (2020)[183], which provides the MITK-based CemrgApp for visualization and manipulation of cardiovascular data, does not follow a generic approach for integrating Docker-based image methods, but only uses the Docker API within the code execution. This was identified by analyzing the project's Github repository[i]. The proposed solution in this thesis focuses on a minimal set of generic programming utilities/classes to integrate new Docker-based (image) processing methods into MITK, thus clearly differentiating itself from this previous work. Moreover, the solution as stand alone external project of MITK does not come with any unwanted overhead.

The presented integration concept is general and open in the way it can be implemented. For this work the interactive application M²aia were chosen for implementation of the concept. The author believes that the concept of integration can be used in other applications, while keeping the Docker-based processing containers unmodified between different interactive applications, reducing the integration efforts overall.

6.2.3 Supporting ImzML-based Deep Learning in Python

pyM²aia was developed as an open-source Python package for processing and analyzing MSI data, with a particular focus on deep learning applications. The package provides researchers working in the field of MSI with a comprehensive toolkit in

[i] https://github.com/OpenHeartDevelopers/CemrgApp; accessed April 2024

Python and enables the processing, visualization, and analysis of large and complex data sets in imzML format. pyM²aia employs the same import routines as M²aia and thus forms an interface to this application. This interface ensures the interchangeability of data elements with M²aia, thus facilitating the exploration and annotation of MSI data and MSI-related image data.

The package thus facilitates memory and computationally efficient loading of imzML datasets and enables the processing of large MSI datasets without excessive memory consumption. This efficiency is crucial for feeding large amounts of data into deep neural networks during training. pyM²aia is optimized for DL tasks and provides batch generators for typical MSI data access strategies, enabling the creation of readable and maintainable DL pipelines. The package is compatible with a number of popular deep learning libraries, including TensorFlow/Keras and PyTorch. The proposed package supports various strategies for handling MSI data:

- Spectral Strategy: Uses spectral information without considering spatial relationships between spectra.
- Spatial Strategy: Focuses on the spatial properties of molecular distributions without intra-spectral relationships.
- Spatio-Spectral Strategy: Utilizes both spatial and spectral information simultaneously, which is computationally demanding but provides comprehensive data analysis.

Supporting these strategies underscores the flexibility of pyM²aia and its potential to advance MSI research.

The presented exemplary applications, such as the generation of ion images and the training of DL models, demonstrate the practical use of pyM²aia. The examples presented not only serve to illustrate the capabilities of the package, but also provide valuable resources for researchers wishing to perform similar analyses. Querying imzML metadata, applying signal processing methods, and generating ion images are fundamental tasks in MSI research, and thus a prerequisite also for new MSI-based deep learning models. In addition, the imzML file format standard for MSI data can be used to provide raw data or, if processed, detailed information about the signal processing steps applied. In contrast to other open standards for the conversion of proprietary file formats from device manufacturers, the handling and provision of this information is already specified by the imzML standard, which is crucial for enhancing reproducibility.

Although pyM²aia facilitates the development of DL approaches for MSI, DL for MSI still faces a number of inherent challenges and limitations. The high computational requirements of DL, coupled with the significant size of MSI datasets, can overwhelm even advanced hardware resources. Furthermore, while pyM²aia provides a robust framework for data processing and DL integration, the quality and consistency of MSI data remain critical factors. Batch effects, fluctuations in data quality and the "curse of dimensionality" are persistent problems in MSI research that cannot be solved independently by pyM²aia. Overcoming these challenges requires a combination of optimized data acquisition techniques, standardized pre-processing protocols and continuous advances in DL methods.

Potential avenues for future development include encouraging contributions from the community to expand the library with example applications and processing functions. Another important direction is the promotion of interdisciplinary collaboration between computer scientists, biologists, and chemists to develop more sophisticated models and applications.

In summary, pyM²aia provides essential tools for efficient data processing, especially in the context of deep learning integration. The evolution of the package promises to remain a valuable resource for the MSI community, driving innovation in both data analytics and deep learning applications.

6.2.4 Answers to the Posed Research Questions

The performed use-cases support the following research questions as posed in chapter 3 *Interactive Multi-Modal 2D/3D MSI Data Analysis*:

- Can an application framework be provided that allows researchers comprehensive interactive access to MSI datasets, simultaneously for multi-modal and 3D MSI?
  - The framework successfully provides comprehensive interactive access to MSI datasets, enabling simultaneous multi-modal and 3D analysis. This is achieved through the development of intuitive visualization and interaction tools that support dynamic exploration of both 2D and 3D MSI data. The framework handles large datasets efficiently, maintaining performance and scalability, which is crucial for responsive data manipulation and visualization. The integration of support for various biomedical image formats ensures broad compatibility, allowing researchers to seamlessly navigate and analyze complex MSI datasets.
- Can this framework be used to implement the lacking capabilities of advanced interactive concepts for multi-modal image fusion and 3D image reconstructions?
  - The framework implements advanced interactive concepts for multi-modal image fusion and 3D image reconstruction. It provides tools for accurate alignment and fusion of images from different modalities, such as combining MSI data with histological images. The image-based registration methods supported by the framework allow precise alignment, and interactive tools enable users to assess and correct registration results dynamically. Additionally, the framework supports both rigid and deformable image registration techniques for creating comprehensive multi-modal image fusions or 3D reconstructions.
- Is it possible to use this framework for common workflows in the field of MSI, such as rapid data exploration and creation of spatial annotations?
  - The framework supports common workflows in MSI, facilitating rapid data exploration and the creation of spatial annotations. It provides efficient data handling capabilities, enabling quick loading, processing, and visualization of MSI datasets. The framework also includes intuitive tools for creating and manipulating spatial annotations, allowing researchers to mark regions of interest and link them to specific molecular features. This streamlines the workflow for researchers, enhancing the usability and efficiency of MSI analysis.
- Is it possible to facilitate the integration and development of the more advanced processes of the molecular analysis workflow, such as the identification of relevant peaks and biomarkers?

  - The framework facilitates the integration and development of advanced molecular analysis processes, including peak identification and biomarker discovery. The framework also supports the integration of custom analytical methods through programming language-independent interfaces, allowing researchers to develop and incorporate their own tools. This flexibility enhances the accuracy and depth of molecular investigations, fostering innovation in MSI research.

- Can fast and memory-efficient processing techniques tailored to the imzML file format enhance the training and development of deep learning models by supporting optimized and convenient data access interfaces?

  - The framework demonstrates that using fast and memory-efficient processing techniques tailored to the imzML file format supports the training and development of deep learning models. The imzML standard, which allows for the storage of raw data and documentation of preprocessing steps within the file format, promotes FAIR (Findable, Accessible, Interoperable, and Reusable) data exchange. By facilitating direct processing of raw data, the framework avoids the pitfalls associated with using preprocessed datasets, such as loss of reproducibility and undocumented preprocessing steps. The optimized data access interfaces ensure that large volumes of MSI data can be efficiently processed, enabling the effective training of deep learning models. This efficiency is crucial for researchers aiming to leverage machine learning techniques to extract meaningful patterns and insights from MSI data.

6.3 A Software Ecosystem for Remote Analysis of MSI Data

The proposed software ecosystem addresses a critical need in biomedical research for efficient, interactive, and remote access to large and complex imaging datasets. The motivation behind this work is rooted in the exponential growth of imaging data and the increasing complexity of image analysis tasks. Traditional methods of data transfer and local processing are becoming increasingly impractical, especially for data-intensive processes such as those involved in MSI. This ecosystem aims to alleviate these challenges by leveraging centralized resources, thereby enhancing collaborative research efforts, and streamlining workflows and development of data analysis methods.

Biomedical imaging is foundational in fields such as pathology, radiology, and cell biology. As imaging techniques continue to evolve, they produce datasets that are not only larger in size but also more detailed and complex. The sheer volume of data necessitates advanced computational resources for storage and processing. The conventional approach of transferring these massive datasets to local systems for analysis is not only inefficient but also impractical for many research settings. This inefficiency can hinder scientific progress, especially when quick and accurate data interpretation is required, as in the multidisciplinary projects like M²Aind and M²OLIE, where timely insights can significantly influence clinical outcomes.

The ecosystem's capabilities are multifaceted, focusing on remote interactive access, scalable architecture, and enhanced collaboration. One of the key features is the ability to explore and analyze biomedical images interactively from remote locations. This capability is crucial for tasks such as image fusion, spatial annotation, and deep learning model development. The interactive nature of the tools allows

researchers to adjust visualization parameters, perform complex image processing tasks, and create detailed annotations without the need for local data storage. This remote access is particularly beneficial for handling hyperspectral and 3D imaging data, which require substantial computational power and sophisticated visualization tools.

The ecosystem's architecture is built on Docker-based containers, ensuring consistency and reliability across different environments. This containerization approach simplifies the deployment process and supports a variety of applications essential for biomedical imaging. For instance, applications like M²aia, QuPath, ilastik, and imageJ are packaged into Docker containers, allowing them to be easily deployed and managed on centralized servers. This approach not only enhances performance and resource utilization but also provides a scalable solution that can accommodate the growing demands of biomedical imaging research.

Integrated Development Environments (IDEs) such as Visual Studio Code (VS Code), JupyterLab, and RStudio play a crucial role in the ecosystem by providing robust platforms for remote development. Among these, VS Code is particularly notable for its extensive support for various programming languages and its efficient remote development capabilities through extensions provided by the Visual Studio Code Remote Development[i] project. The Application Controller utilities[ii] integrated into VS Code further enhance the development workflow by providing rapid access to remote interactive image processing applications and streamlining the management of these interactive applications.

The ecosystem also fosters enhanced collaboration among researchers. By allowing multiple users to work simultaneously on the same datasets without conflicts, it promotes interdisciplinary collaboration and ensures that diverse expertise can be integrated into the research process. This collaborative capability is particularly important in multidisciplinary projects where the integration of different perspectives can lead to more comprehensive and accurate data interpretations. Furthermore, by keeping datasets on centralized servers and providing remote access via web interfaces, the ecosystem ensures data security and integrity, preventing risks associated with data corruption or loss during transfers.

A comparison between the proposed ecosystem and Galaxy[1], the Kaapana[iii]/Joint Imaging Platform (JIP)[2], OMERO[98], and OpenMSI[3] highlights several distinct differences in design, functionality, and overall objectives. While both platforms aim to provide robust solutions for biomedical imaging, they cater to different needs and offer unique features.

1. Interactive Remote Access and Scalability: a major strength of the proposed ecosystem is its provision of interactive remote access to imaging datasets. This allows researchers to explore and analyze images interactively from remote locations, enabling tasks such as image fusion, spatial annotation, and deep learning model development without the need for local data storage. The ecosystem's tools support complex image processing tasks and enhance the handling of hyperspectral and 3D imaging data. This level of interactivity is crucial for responsive data exploration and immediate feedback, which are essential for agile research and development practices.

    - In contrast, JIP is primarily focused on providing a centralized platform for managing and processing biomedical imaging data and workflows.

[i] https://github.com/Microsoft/vscode-remote-release; accessed April 2024
[ii] https://m2aia.github.io/m2aia; accessed April 2024
[iii] https://www.kaapana.ai/; accessed April 2024

While it supports remote access to some extent, its primary emphasis is on creating reproducible processing pipelines rather than interactive exploration. JIP's architecture is designed to integrate various imaging modalities and support standardized workflows.

- In contrast, the Galaxy platform focuses on providing a robust, reproducible, and scalable environment for data analysis and workflow management. Galaxy is well-known for its web-based interface that allows users to create, share, and execute data analysis workflows without needing extensive programming knowledge. It excels in managing and processing large-scale data through a web interface and supports a wide range of bioinformatics tools and applications. However, its primary focus is on batch processing and reproducibility rather than interactive data exploration.
- OpenMSI also supports remote access to 2D MSI data, which are stored in HDF5 format, but with a specific focus on providing a web platform for data visualization and a Python-based interface to platform internally stored MSI data. OpenMSI enables fast and convenient access to the platform integrated MSI data, metadata, and derived analysis results stored remotely. It provides web-based tools for data sharing, visualization, and analysis, ensuring that large MSI datasets can be accessed and processed efficiently. However, OpenMSI primarily emphasizes batch processing and basic visualization rather than the level of ad-hoc interactivity and immediate feedback offered by the proposed ecosystem.
- OMERO, in contrast, is designed primarily for managing and sharing large sets of scientific image data. It provides robust data management capabilities, including the storage, retrieval, and sharing of diverse imaging data across various modalities. While OMERO supports remote access, its primary focus is on data management and integration rather than on interactive exploration. OMERO excels in providing a centralized repository for image data, facilitating data sharing, collaboration, and long-term storage.

2. Docker-Based Containers and Application Flexibility: the proposed ecosystem employs Docker-based containers to ensure consistency and reliability across different environments. This containerization approach simplifies deployment and supports a wide range of applications essential for biomedical imaging, such as M²aia, QuPath, ilastik, and imageJ. These applications are packaged into Docker containers, allowing them to be easily deployed and managed on centralized servers. This approach enhances performance, resource utilization, and scalability, making the ecosystem adaptable to the growing demands of biomedical research. The proposed ecosystem, with its focus on interactive applications and remote data processing, provides a flexible environment for tasks requiring direct interaction with data.

    - JIP also utilizes containerization but focuses more on providing a stable and reproducible environment for data processing workflows. While JIP supports a range of applications, its emphasis is on maintaining standardized processing pipelines and ensuring reproducibility across different datasets and studies. This focus on reproducibility is crucial for ensuring the integrity of biomedical research.

- Galaxy also utilizes containerization and virtualization technologies like Docker and Kubernetes to ensure scalability and reproducibility. However, Galaxy's emphasis is on creating a seamless user experience for running standardized workflows. While it supports a broad array of tools and allows for the integration of custom tools, the flexibility of deploying and managing interactive applications is not its primary strength.
- OpenMSI, while not specifically focused on containerization, provides a robust platform for managing and analyzing MSI data. It integrates various tools for data processing and visualization but does not emphasize the same level of flexibility in deploying and managing interactive applications as the proposed ecosystem.
- OMERO also supports the use of containerization technologies but focuses on providing a stable and scalable environment for data management rather than for deploying interactive applications. OMERO's architecture is designed to integrate with various image analysis tools and workflows, providing a centralized platform for managing and sharing imaging data. While OMERO can integrate with analysis tools, its primary emphasis is on data storage, metadata management, and ensuring data integrity across different projects and studies.

3. Integrated Development Environments and Developer Support: the proposed ecosystem leverages Integrated Development Environments (IDEs) such as Visual Studio Code (VS Code), JupyterLab, and RStudio for remote development. VS Code, in particular, offers extensive support for various programming languages and efficient remote development capabilities through extensions like Remote SSH and Remote Tunnels. The ecosystem also includes Application Controller utilities that enhance the development workflow by providing rapid access to remote image data and streamlining the management of interactive applications. The proposed ecosystem's emphasis on integrating with widely-used IDEs and providing robust developer tools makes it more versatile and user-friendly for researchers who need to develop and test new image processing methods remotely.

    - JIP, while providing tools for remote development, primarily focuses on web-based interfaces and platforms like the Galaxy Project for creating and managing data processing workflows. These tools are highly effective for creating reproducible pipelines but may not offer the same level of integration with popular IDEs or the extensive developer support found in the proposed ecosystem.
    - Galaxy, while providing a comprehensive environment for creating and managing workflows, does not focus on integrating popular IDEs for code development. Galaxy's strength lies in its ability to simplify the process of running complex bioinformatics analyses through a graphical interface, making it accessible to researchers with limited programming skills.
    - OpenMSI focuses more on providing a platform for analyzing and visualizing MSI data rather than integrating popular IDEs for code development. It offers a web-based interface that simplifies the process of accessing and analyzing MSI data but does not provide the same level of integration with development environments that support extensive coding and method development.

- OMERO, while providing extensive support for image data management, does not focus on integrating IDEs for code development. OMERO's strength lies in its ability to manage large-scale image data and metadata, providing tools for data annotation, tagging, and linking with experimental metadata.

4. Enhanced Collaboration and Data Security: the proposed software ecosystem is designed to enhance collaboration by allowing multiple users to work simultaneously on the same datasets without conflicts. This capability promotes interdisciplinary collaboration and ensures that diverse expertise can be integrated into the research process. The ecosystem also ensures data security and integrity by keeping datasets on centralized servers and providing remote access via web interfaces, preventing risks associated with data corruption or loss during transfers. The proposed ecosystem's emphasis on interactivity and immediate feedback makes it more suitable for collaborative research environments where rapid iteration and real-time data sharing are crucial

   - JIP also supports collaborative efforts but emphasizes standardized workflows and reproducibility to ensure data integrity across studies. While JIP provides tools for data sharing and collaboration, its primary focus is on creating reproducible pipelines rather than facilitating real-time interactive collaboration.
   - Galaxy also supports collaboration through its platform, allowing users to share workflows, histories, and datasets easily. It provides a reproducible environment where analyses can be shared and rerun by different users, ensuring transparency and consistency in research. However, Galaxy's collaboration features are geared towards sharing reproducible workflows rather than facilitating real-time interactive collaboration on image data.
   - OpenMSI also supports collaboration by providing a centralized platform for accessing and sharing MSI data and analysis results. It enables researchers to share datasets and results through its web interface, ensuring that data can be collaboratively analyzed and interpreted. However, OpenMSI's primary focus is on providing data access and visualization of MSI data rather than facilitating real-time interactive collaboration.
   - OMERO also supports collaboration by providing a centralized repository for imaging data, enabling researchers to share datasets, annotations, and analysis results. OMERO's data management capabilities facilitate collaboration by ensuring that data is consistently organized and easily accessible to all members of a research team. However, OMERO's focus is more on the management and sharing of data rather than on providing interactive tools for real-time collaboration.

5. User Accessibility and Administrative Overhead: The proposed ecosystem is designed to be user-friendly with minimal administrative and learning overhead. This design ensures that researchers with varying levels of technical expertise can easily adopt and use the platform. The integration of remote execution, development, and interactive capabilities within a scalable and user-friendly framework positions this ecosystem as a valuable tool for a wide range of biomedical research applications. The proposed ecosystem's flexibility and ease of use make it more accessible for researchers who require a dynamic and adaptable platform. With its Docker-based deployment and robust developer

support, the ecosystem aims to reduce overhead and provide a more seamless experience for end-users.

- JIP, while user-friendly, may require more initial setup and configuration, particularly for integrating new tools or adapting existing workflows. The focus on reproducibility and standardization in JIP can sometimes lead to a more rigid framework, which may not be as easily adaptable to the specific needs of individual researchers or projects.
- Galaxy is also known for its user-friendly interface, which allows researchers to perform complex analyses without needing extensive bioinformatics training. Its web-based platform reduces the need for local software installation and simplifies the management of bioinformatics tools. However, setting up and maintaining a Galaxy instance, especially for large-scale data processing, can require significant administrative effort.
- OpenMSI is also user-friendly, particularly in its provision of a web-based platform that simplifies the access and analysis of MSI data. However, setting up and maintaining an OpenMSI instance, particularly for large-scale data processing, can require significant administrative effort. The proposed ecosystem, with its Docker-based deployment and robust developer support, aims to reduce this overhead and provide a more seamless experience for end-users.
- OMERO is also user-friendly, particularly in its provision of a centralized data management platform that simplifies the storage, retrieval, and sharing of imaging data. However, setting up and maintaining an OMERO instance can require significant administrative effort, especially when integrating with external analysis tools or adapting to specific research workflows.

The proposed software ecosystem, Joint Imaging Platform (JIP), Galaxy, OMERO, and OpenMSI each provide distinct advantages for biomedical imaging data management and analysis. The proposed ecosystem excels in interactive remote access, flexibility through Docker-based containers, and robust developer support, making it ideal for real-time data exploration and agile research practices. JIP focuses on creating reproducible workflows and integrating various imaging modalities, offering stability and consistency for data processing. Galaxy simplifies complex bioinformatics analyses through a web interface, emphasizing reproducibility and accessibility for researchers with limited programming skills. OMERO provides comprehensive data management and integration capabilities, ensuring organized, searchable, and shareable imaging data across projects. OpenMSI specializes in high-performance data processing and visualization for MSI data, facilitating efficient access and analysis.

In the opinion of the author, the potential impacts of this software ecosystem are significant. It promises to enhance the efficiency and speed of biomedical research by enabling rapid iteration and testing of new image processing methods. This agility is crucial for advancing scientific discoveries and innovations. Moreover, the ecosystem minimizes the need for large data transfers, thereby reducing time and resource consumption and allowing researchers to focus more on analysis and method development. The user-friendly design and minimal administrative overhead ensure that the ecosystem is accessible to researchers with varying levels of technical expertise, making it a valuable tool across the biomedical research community. Due to its flexibility in how to deploy and utilize the proposed tools, the integration of

individual components or a concurrent usage of the proposed system with other systems, as those described above, is possible and desirable.

In conclusion, the proposed software ecosystem represents a substantial advancement in the field of biomedical imaging. By leveraging remote resources and providing interactive tools, it addresses the critical challenges posed by large and complex datasets, particularly in MSI. The ecosystem not only enhances the efficiency of data analysis but also fosters interdisciplinary collaboration and innovation, ultimately contributing to the advancement of biomedical research and clinical practice. The integration of remote development, execution, and interactive capabilities within a scalable and user-friendly framework positions this ecosystem as a pivotal tool for future biomedical research endeavors.

6.3.1 Answers to the Posed Research Questions

The performed use-cases support the following research questions as posed in chapter 4 *Concepts for Interactive Remote Working*:

- Can interactive applications, like M²aia, be hosted on remote resources to process image data for which there are no interactive remote solutions yet, e.g., MSI data?
  - Yes, interactive applications like M²aia can indeed be hosted on remote resources to process image data, including MSI data, for which there are currently no comprehensive interactive remote solutions. The proposed software ecosystem leverages Docker-based containerization to encapsulate these interactive applications, ensuring consistency and reliability across various environments. By hosting M²aia on remote servers, the ecosystem enables researchers to interact with complex MSI datasets remotely, providing tools for visualization, processing, and analysis that are typically limited to local systems. This approach not only addresses the lack of existing interactive remote solutions for MSI data but also enhances accessibility and usability, allowing researchers to perform sophisticated image analysis tasks without the need for extensive local computational resources.
- Can interactive image exploration and thereby the development process of new image-based processing methods be supported or/and completely shifted to remote resources?
  - The interactive image exploration and development process of new image-based processing methods can be fully supported and even completely shifted to remote resources. The proposed ecosystem integrates powerful Integrated Development Environments (IDEs) such as Visual Studio Code (VS Code), JupyterLab, and RStudio, which are tailored for remote development. These IDEs provide robust tools for coding, debugging, and deploying new image processing methods directly on centralized servers. Additionally, the ecosystem includes Application Controller utilities that facilitate the management of remote applications and streamline the development workflow. This setup allows researchers to iteratively test and refine their algorithms in a real-world setting, receiving immediate feedback on their performance. By providing interactive access to image data and enabling remote execution of development tasks, it is expected that the

ecosystem significantly accelerates the pace of innovation and enhances the efficiency of the development process.

- Can remote interactive access to image processing applications be realized in a simple, efficient, and user-friendly way?
  - Interactive access to image processing applications can indeed be realized in a simple, efficient, and user-friendly way through the proposed software ecosystem. By employing Docker-based containers, the ecosystem simplifies the deployment and management of interactive applications, ensuring that they are accessible via standard web browsers. This approach eliminates the need for complex installations and configurations on local systems, making it easier for users to interact with the applications. The ecosystem is designed with minimal administrative overhead and includes intuitive interfaces that cater to researchers with varying levels of technical expertise. The integration with popular IDEs and the provision of robust developer tools further enhance the user experience, allowing researchers to focus on their scientific tasks without being burdened by technical complexities. Overall, the ecosystem provides a seamless and accessible environment for remote interactive image processing, making advanced biomedical research more accessible and efficient.

Chapter 7

Summary

In many areas of biomedical research, images are crucial for scientific progress. Interactive access to these images is essential, enhancing understanding and facilitating advancements, particularly in fields like pathology, radiology, and cellular biology. As imaging techniques continue to advance, generating ever more detailed datasets, the amount of data to be stored and processed will continue to grow. Consequently, data and computationally intensive processes are being increasingly relocated to centralized resources with substantial storage and processing capabilities. However, large, multidimensional and multi-modal biomedical images, such as those generated in experiments with mass spectrometry imaging, pose a major challenge for fast, comprehensive and interactive remote access. Processes as image data exploration, image analysis, the development of new image analysis methods, and interdisciplinary collaboration of domain experts can be hampered if data-intensive transfers to local systems are required, e.g. for processing of images with the interactive software the domain expert is familiar with. Current efforts to utilize remote resources focus on providing integrated environments for remote development and applications for execution of reproducible image analysis, while lacking comprehensive interactive capabilities to work with high-dimensional image data. Due to the increased complexity of remotely executed image related workflows and related research problems, re-usability and interoperability of solutions in the form of software components becomes essential.

The primary objective of this thesis was to develop concepts for interdisciplinary research that facilitate remote image analysis and remote development in the field of MSI data processing and beyond. The proposed concepts form a software ecosystem of interacting components.

In the following paragraphs, the main contributions of this thesis are summarized.

A Novel Openly-accessible Interactive Application for Processing of MSI Datasets

A requirements analysis and review of established technologies lead to the design and creation of the interactive application M²aia, which stands for Mass spectrometry Imaging Application for Interactive Analysis in MITK. It is a novel C++ based modular extension of the MITK framework and designed for the exploration and processing of multi-modal 2D/3D MSI datasets in the imzML file format. It enables users to realize customizable workflows and provides fast and memory-efficient MSI data processing, such as signal processing and ion-image generation, realized as multiple plug-in views. A special focus is on providing interactive capabilities to image-based registration concepts for 3D image reconstruction and multi-modal image fusion. Since image-based registration is highly sensitive to initial configuration, qualitative and quantitative registration-result evaluation and interactive correction strategies were implemented. To fully support multi-modal workflows, strategies have been

developed to transfer annotations from structure-rich optical images to the molecular images captured by MSI.

The features-rich interactive capabilities of M²aia were demonstrated by two use-cases. A 3D reconstruction of a multi-modal mouse brain dataset demonstrates the efficiency of importing data in imzML format and shows the applicability of the proposed image-based registration strategies. Additionally, a use-case demonstrating a biomarker identification procedure was realized and compared to reference data, showing the validity of the implemented signal processing methods. Performance tests showed the efficiency of the import of multiple large MSI data in imzML format.

Programming Language Independent Integration of Third-party Applications

A novel strategy for the integration of third-party image processing methods into interactive environments was proposed and successfully demonstrated. It is based on Docker and enables the programming language-independent integration of image processing methods. This can be realized fast and flexible with a high degree of freedom due to the Docker based containerization. It allows researchers the creation of highly customized image processing applications and workflows according to their needs.

A proof-of-concept for the integration of third-party image processing methods was demonstrated by integrating cutting-edge technologies into the interactive application M²aia. The integration of the Deep Learning and Python-based *TotalSegmentator* for segmentation of clinical images as well as, for MSI datasets, the *UMAP* dimensionality reduction, the *moleculaR* R package for molecular probabilistic maps, and the pyM²aia based unsupervised peak learning were realized.

A Novel Python Package for the Development Support of Deep Learning Applications in MSI

Based on the import capabilities of M²aia, a Python package (pyM²aia) has been designed and implemented to support the development and creation of new Deep Learning applications in the field of MSI. To facilitate efficient data access to MSI datasets, the spectral, spatial, and spatio-spectral strategies were defined and implemented as batch generation utilities. Extensive Deep Learning example applications showed the applicability and validity of the implemented strategies.

The importation of datasets was conducted according to the imzML standard, a widely recognized and adopted file format. This approach contrasts sharply with the less transparent, in-house data conversion methods that often lead to the creation of potentially altered datasets with undocumented processing steps. By using data in imzML format, the reproducibility of experiments is enhanced, as it standardizes the data handling process across different studies and labs.

A Software Ecosystem for the Remote Analysis of MSI Data

Novel strategies for remote interaction with image data have been implemented with a focus on remote processing of multi-modal MSI datasets. Special attention was paid to providing a flexible system for the development of new image-based processing methods, while at the same time facilitating collaboration by access to remote image data collections. A containerization strategy was developed and several image processing applications were packaged for remote interactive access using a standard web browser. A container management extension was designed and implemented to facilitate the use of application containers during development.

To demonstrate the capabilities of the proposed software ecosystem for remote multimodal image data analysis, several use cases were realized, two of which were published in two co-authored articles. For this purpose, several interactive applications such as MITK, M²aia, fiji/imageJ, and ilastik were packaged in Docker images, providing a diverse set of tools for remote interactive analysis in multimodal research questions. Specifically, focusing on the remote development cycle for image processing method, the system provides exceptional flexibility and usability of rapid handling of remote MSI and biomedical imaging data.

Designing individual workflows on remote or centralized resources demonstrated the dynamic and adaptable capabilities of the software ecosystem for remote analysis and development. The Application Controller utilities facilitate the management of individual interactive applications in a way that is both user- and developer-friendly. The lightweight set of software components requires a minimal effort for installation and administration, which is a clear advantage over comparable systems.

In conclusion, the software ecosystem for remote analysis and development, as presented in this thesis, has demonstrated substantial extensibility and flexibility, making it highly suitable for executing complex MSI and image-related workflows. Given its open-source nature, this ecosystem not only facilitates widespread adaptation and customization by researchers but also promotes collaborative enhancements and innovations. These characteristics significantly enhance its applicability across diverse research domains, thereby justifying the expectation of a high impact on future research projects.

Bibliography

1. Afgan, E., Baker, D., Batut, B., van den Beek, M., Bouvier, D., Čech, M., Chilton, J., Clements, D., Coraor, N., Grüning, B. A., *et al.,* (2018), The Galaxy Platform for Accessible, Reproducible and Collaborative Biomedical Analyses: 2018 Update, *Nucleic Acids Research,* 46: W537–W544, doi:10.1093/nar/gky379
2. Scherer, J., Nolden, M., Kleesiek, J., Metzger, J., Kades, K., Schneider, V., Bach, M., Sedlaczek, O., Bucher, A. M., Vogl, T. J., *et al.,* (2020), Joint Imaging Platform for Federated Clinical Data Analytics, *JCO Clinical Cancer Informatics:* 12, doi:10.1200/CCI.20.00045
3. Rübel, O., Greiner, A., Cholia, S., Louie, K., Bethel, E. W., Northen, T. R. & Bowen, B. P., (2013), OpenMSI: A High-Performance Web-Based Platform for Mass Spectrometry Imaging, *Analytical Chemistry,* 85: 10354–10361, doi:10.1021/ac402540a
4. Menegidio, F. B., Jabes, D. L., Costa De Oliveira, R. & Nunes, L. R., (2018), Dugong: A Docker Image, Based on Ubuntu Linux, Focused on Reproducibility and Replicability for Bioinformatics Analyses, *Bioinformatics,* 34: 514–515, doi:10.1093/bioinformatics/btx554
5. Krestensen, K. K., Heeren, R. M. A. & Balluff, B., (2023), State-of-the-Art Mass Spectrometry Imaging Applications in Biomedical Research, *The Analyst,* doi:10.1039/d3an01495a
6. Balluff, B., Heeren, R. M. & Race, A. M., (2021), An Overview of Image Registration for Aligning Mass Spectrometry Imaging with Clinically Relevant Imaging Modalities, *Journal of Mass Spectrometry and Advances in the Clinical Lab:* S2667145X21000353, doi:10.1016/j.jmsacl.2021.12.006
7. Patterson, N. H., Doonan, R. J., Daskalopoulou, S. S., Dufresne, M., Lenglet, S., Montecucco, F., Thomas, A. & Chaurand, P., (2016), Three-Dimensional Imaging MS of Lipids in Atherosclerotic Plaques: Open-source Methods for Reconstruction and Analysis, *PROTEOMICS,* 16: 1642–1651, doi:10.1002/pmic.201500490
8. Kriegsmann, J., Kriegsmann, M. & Casadonte, R., (2015), MALDI TOF Imaging Mass Spectrometry in Clinical Pathology: A Valuable Tool for Cancer Diagnostics (Review), *International Journal of Oncology,* 46: 893–906, doi:10.3892/ijo.2014.2788
9. Basu, S. S., Regan, M. S., Randall, E. C., Abdelmoula, W. M., Clark, A. R., Gimenez-Cassina Lopez, B., Cornett, D. S., Haase, A., Santagata, S. & Agar, N. Y. R., (2019), Rapid MALDI Mass Spectrometry Imaging for Surgical Pathology, *npj Precision Oncology,* 3: 17, doi:10.1038/s41698-019-0089-y
10. Dexter, A., Tsikritsis, D., Belsey, N. A., Thomas, S. A., Venton, J., Bunch, J. & Romanchikova, M., (2022), Next Generation Digital Pathology: Emerging Trends and Measurement Challenges for Molecular Pathology, *Journal of Molecular Pathology,* 3: 168–181, doi:10.3390/jmp3030014

11. Cole, L. M. & Clench, M. R., (2015), Mass Spectrometry Imaging for the Proteomic Study of Clinical Tissue, *PROTEOMICS – Clinical Applications,* 9: 335–341, doi:10.1002/prca.201400103

12. Caprioli, R. M., Farmer, T. B. & Gile, J., (1997), Molecular Imaging of Biological Samples: Localization of Peptides and Proteins Using MALDI-TOF MS, *Analytical Chemistry,* 69: 4751–4760, doi:10.1021/ac970888i

13. Heeren, R. M. A., Smith, D. F., Stauber, J., Kükrer-Kaletas, B. & MacAleese, L., (2009), Imaging Mass Spectrometry: Hype or Hope?, *Journal of the American Society for Mass Spectrometry,* 20: 1006–1014, doi:10.1016/j.jasms.2009.01.011

14. McDonnell, L. A. & Heeren, R. M., (2007), Imaging Mass Spectrometry, *Mass Spectrometry Reviews,* 26: 606–643, doi:10.1002/mas.20124

15. Eriksson, C., Masaki, N., Yao, I., Hayasaka, T. & Setou, M., (2013), MALDI Imaging Mass Spectrometry—A Mini Review of Methods and Recent Developments, *Mass Spectrometry,* 2: S0022, doi:10.5702/massspectrometry.S0022

16. Petras, D., Jarmusch, A. K. & Dorrestein, P. C., (2017), From Single Cells to Our Planet—Recent Advances in Using Mass Spectrometry for Spatially Resolved Metabolomics, *Current Opinion in Chemical Biology,* 36: 24–31, doi:10.1016/j.cbpa.2016.12.018

17. Buchberger, A. R., DeLaney, K., Johnson, J. & Li, L., (2018), Mass Spectrometry Imaging: A Review of Emerging Advancements and Future Insights, *Analytical Chemistry,* 90: 240–265, doi:10.1021/acs.analchem.7b04733

18. Ràfols, P., Vilalta, D., Brezmes, J., Cañellas, N., del Castillo, E., Yanes, O., Ramírez, N. & Correig, X., (2016), Signal Preprocessing, Multivariate Analysis and Software Tools for MA(LDI)-TOF Mass Spectrometry Imaging for Biological Applications, *Mass Spectrometry Reviews,* 37: 281–306, doi:10.1002/mas.21527

19. Maloof, K. A., Reinders, A. N. & Tucker, K. R., (2020), Applications of Mass Spectrometry Imaging in the Environmental Sciences, *Current Opinion in Environmental Science & Health,* 18: 54–62, doi:10.1016/j.coesh.2020.07.005

20. Alexandrov, T., (2020), Spatial Metabolomics and Imaging Mass Spectrometry in the Age of Artificial Intelligence, *Annual Review of Biomedical Data Science,* 3: 61–87, doi:10.1146/annurev-biodatasci-011420-031537

21. Piehowski, P. D., Zhu, Y., Bramer, L. M., Stratton, K. G., Zhao, R., Orton, D. J., Moore, R. J., Yuan, J., Mitchell, H. D., Gao, Y., *et al.,* (2020), Automated Mass Spectrometry Imaging of over 2000 Proteins from Tissue Sections at 100-*M*m Spatial Resolution, *Nature Communications,* 11: 8, doi:10.1038/s41467-019-13858-z

22. Dueñas, M. E., Larson, E. A. & Lee, Y. J., (2019), Toward Mass Spectrometry Imaging in the Metabolomics Scale: Increasing Metabolic Coverage Through Multiple On-Tissue Chemical Modifications, *Frontiers in Plant Science,* 10

23. Robichaud, G., Barry, J. A. & Muddiman, D. C., (2014), IR-MALDESI Mass Spectrometry Imaging of Biological Tissue Sections Using Ice as a Matrix, *Journal of the American Society for Mass Spectrometry,* 25: 319–328, doi:10.1007/s13361-013-0787-6

24. Siuzdak, G., (2023), Subcellular Quantitative Imaging of Metabolites at the Organelle Level, *Nature Metabolism,* 5: 1446–1448, doi:10.1038/s42255-023-00882-z

25. Tuck, M., Grélard, F., Blanc, L. & Desbenoit, N., (2022), MALDI-MSI Towards Multimodal Imaging: Challenges and Perspectives, *Frontiers in Chemistry,* 10: 904688, doi:10.3389/fchem.2022.904688

26. Geier, B., Oetjen, J., Ruthensteiner, B., Polikarpov, M., Gruber-Vodicka, H. R. & Liebeke, M., (2021), Connecting Structure and Function from Organisms to Molecules in Small-Animal Symbioses through Chemo-Histo-Tomography, *Proceedings of the National Academy of Sciences,* 118, doi:10.1073/pnas.2023773118

27. Patterson, N. H., Yang, E., Kranjec, E.-A. & Chaurand, P., (2019), Co-Registration and Analysis of Multiple Imaging Mass Spectrometry Datasets Targeting Different Analytes, *Bioinformatics,* 35: 1261–1262, doi:10.1093/bioinformatics/bty780

28. Chen, M., Tustison, N. J., Jena, R. & Gee, J. C. in *Machine Learning for Brain Disorders* (ed Colliot, O.) 435–458 (Springer US, New York, NY, 2023) Accessed 21-3-2024

29. Oliveira, F. P. & Tavares, J. M. R., (2014), Medical Image Registration: A Review, *Computer Methods in Biomechanics and Biomedical Engineering,* 17: 73–93, doi:10.1080/10255842.2012.670855

30. Seeley, E. H. & Caprioli, R. M., (2012), 3D Imaging by Mass Spectrometry: A New Frontier, *Analytical Chemistry,* 84: 2105–2110, doi:10.1021/ac2032707

31. Palmer, A. D. & Alexandrov, T., (2015), Serial 3D Imaging Mass Spectrometry at Its Tipping Point, *Analytical Chemistry,* 87: 4055–4062, doi:10.1021/ac504604g

32. Thiele, H., Heldmann, S., Trede, D., Strehlow, J., Wirtz, S., Dreher, W., Berger, J., Oetjen, J., Kobarg, J. H., Fischer, B., *et al.,* (2014), 2D and 3D MALDI-imaging: Conceptual Strategies for Visualization and Data Mining, *Biochimica et Biophysica Acta (BBA) - Proteins and Proteomics,* 1844: 117–137, doi:10.1016/j.bbapap.2013.01.040

33. Oetjen, J., Veselkov, K., Watrous, J., McKenzie, J. S., Becker, M., Hauberg-Lotte, L., Kobarg, J. H., Strittmatter, N., Mróz, A. K., Hoffmann, F., *et al.,* (2015), Benchmark Datasets for 3D MALDI- and DESI-imaging Mass Spectrometry, *GigaScience,* 4, doi:10.1186/s13742-015-0059-4

34. Abdelmoula, W. M., Pezzotti, N., Hölt, T., Dijkstra, J., Vilanova, A., McDonnell, L. A. & Lelieveldt, B. P. F., (2018), Interactive Visual Exploration of 3D Mass Spectrometry Imaging Data Using Hierarchical Stochastic Neighbor Embedding Reveals Spatiomolecular Structures at Full Data Resolution, *Journal of Proteome Research,* 17: 1054–1064, doi:10.1021/acs.jproteome.7b00725

35. Rohlfing, T., (2012), Image Similarity and Tissue Overlaps as Surrogates for Image Registration Accuracy: Widely Used but Unreliable, *IEEE Transactions on Medical Imaging,* 31: 153–163, doi:10.1109/TMI.2011.2163944

36. Balluff, B., Hopf, C., Porta Siegel, T., Grabsch, H. I. & Heeren, R. M. A., (2021), Batch Effects in MALDI Mass Spectrometry Imaging, *Journal of the American Society for Mass Spectrometry,* 32: 628–635, doi:10.1021/jasms.0c00393

37. Van de Plas, R., Yang, J., Spraggins, J. & Caprioli, R. M., (2015), Image Fusion of Mass Spectrometry and Microscopy: A Multimodality Paradigm for Molecular Tissue Mapping, *Nature Methods,* 12: 366–372, doi:10.1038/nmeth.3296

38. Granborg, J. R., Handler, A. M. & Janfelt, C., (2022), Mass Spectrometry Imaging in Drug Distribution and Drug Metabolism Studies – Principles, Applications and Perspectives, *TrAC Trends in Analytical Chemistry,* 146: 116482, doi:10.1016/j.trac.2021.116482

39. Chaurand, P., (2012), Imaging Mass Spectrometry of Thin Tissue Sections: A Decade of Collective Efforts, *Journal of Proteomics,* 75: 4883–4892, doi:10.1016/j.jprot.2012.04.005

40. Takats, Z., (2004), Mass Spectrometry Sampling Under Ambient Conditions with Desorption Electrospray Ionization, *Science,* 306: 471–473, doi:10.1126/science.1104404

41. Leopold, J., Popkova, Y., Engel, K. & Schiller, J., (2018), Recent Developments of Useful MALDI Matrices for the Mass Spectrometric Characterization of Lipids, *Biomolecules,* 8: 173, doi:10.3390/biom8040173

42. Trim, P. J. & Snel, M. F., (2016), Small Molecule MALDI MS Imaging: Current Technologies and Future Challenges, *Methods (San Diego, Calif.),* 104: 127–141, doi:10.1016/j.ymeth.2016.01.011

43. van Agthoven, M. A., Lam, Y. P. Y., O'Connor, P. B., Rolando, C. & Delsuc, M.-A., (2019), Two-Dimensional Mass Spectrometry: New Perspectives for Tandem Mass Spectrometry, *European Biophysics Journal,* 48: 213–229, doi:10.1007/s00249-019-01348-5

44. Karas, M. & Krüger, R., (2003), Ion Formation in MALDI: The Cluster Ionization Mechanism, *Chemical Reviews,* 103: 427–440, doi:10.1021/cr010376a

45. Murray, K. K., (2022), Resolution and Resolving Power in Mass Spectrometry, *Journal of the American Society for Mass Spectrometry,* 33: 2342–2347, doi:10.1021/jasms.2c00216

46. Vestal, M. L., (2009), Modern MALDI Time-of-Flight Mass Spectrometry, *Journal of Mass Spectrometry,* 44: 303–317, doi:10.1002/jms.1537

47. Bowman, A. P., Blakney, G. T., Hendrickson, C. L., Ellis, S. R., Heeren, R. M. A. & Smith, D. F., (2020), Ultra-High Mass Resolving Power, Mass Accuracy, and Dynamic Range MALDI Mass Spectrometry Imaging by 21-T FT-ICR MS, *Analytical Chemistry,* 92: 3133–3142, doi:10.1021/acs.analchem.9b04768

48. Goh, W. W. B., Wang, W. & Wong, L., (2017), Why Batch Effects Matter in Omics Data, and How to Avoid Them, *Trends in Biotechnology,* 35: 498–507, doi:10.1016/j.tibtech.2017.02.012

49. Janowczyk, A., Zuo, R., Gilmore, H., Feldman, M. & Madabhushi, A., (2019), HistoQC: An Open-Source Quality Control Tool for Digital Pathology Slides, *JCO Clinical Cancer Informatics,* 3: CCI.18.00157, doi:10.1200/CCI.18.00157

50. Ly, A., Longuespée, R., Casadonte, R., Wandernoth, P., Schwamborn, K., Bollwein, C., Marsching, C., Kriegsmann, K., Hopf, C., Weichert, W., *et al.,* (2019), Site-to-Site Reproducibility and Spatial Resolution in MALDI–MSI of Peptides from Formalin-Fixed Paraffin-Embedded Samples, *Proteomics. Clinical Applications,* 13: 1800029, doi:10.1002/prca.201800029

51. Kompauer, M., Heiles, S. & Spengler, B., (2017), Autofocusing MALDI Mass Spectrometry Imaging of Tissue Sections and 3D Chemical Topography of Nonflat Surfaces, *Nature Methods,* 14: 1156–1158, doi:10.1038/nmeth.4433

52. Kompauer, M., Heiles, S. & Spengler, B., (2017), Atmospheric Pressure MALDI Mass Spectrometry Imaging of Tissues and Cells at 1.4-*M*m Lateral Resolution, *Nature Methods,* 14: 90–96, doi:10.1038/nmeth.4071

53. Schramm, T., Hester, Z., Klinkert, I., Both, J.-P., Heeren, R. M., Brunelle, A., Laprévote, O., Desbenoit, N., Robbe, M.-F., Stoeckli, M., *et al.,* (2012), imzML — A Common Data Format for the Flexible Exchange and Processing of Mass Spectrometry Imaging Data, *Journal of Proteomics,* 75: 5106–5110, doi:10.1016/j.jprot.2012.07.026

54. Hu, H. & Laskin, J., (2022), Emerging Computational Methods in Mass Spectrometry Imaging, *Advanced Science,* 9: 2203339, doi:10.1002/advs.202203339

55. Savitzky, A. & Golay, M. J. E., (1964), Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry,* 36: 1627–1639, doi:10.1021/ac60214a047

56. Hu, H., Yin, R., Brown, H. M. & Laskin, J., (2021), Spatial Segmentation of Mass Spectrometry Imaging Data by Combining Multivariate Clustering and Univariate Thresholding, *Analytical chemistry,* 93: 3477–3485, doi:10.1021/acs.analchem.0c04798

57. Ovchinnikova, K., Stuart, L., Rakhlin, A., Nikolenko, S. & Alexandrov, T., (2020), ColocML: Machine Learning Quantifies Co-Localization between Mass Spectrometry Images, *Bioinformatics,* 36: 3215–3224, doi:10.1093/bioinformatics/btaa085

58. Hu, H., Bindu, J. P. & Laskin, J., (2022), Self-Supervised Clustering of Mass Spectrometry Imaging Data Using Contrastive Learning, *Chemical Science,* 13: 90–98, doi:10.1039/D1SC04077D

59. Palmer, A., Phapale, P., Chernyavsky, I., Lavigne, R., Fay, D., Tarasov, A., Kovalev, V., Fuchser, J., Nikolenko, S., Pineau, C., *et al.,* (2017), FDR-controlled Metabolite Annotation for High-Resolution Imaging Mass Spectrometry, *Nature Methods,* 14: 57–60, doi:10.1038/nmeth.4072

60. Sementé, L., Baquer, G., García-Altares, M., Correig-Blanchar, X. & Ràfols, P., (2021), rMSIannotation: A Peak Annotation Tool for Mass Spectrometry Imaging Based on the Analysis of Isotopic Intensity Ratios, *Analytica Chimica Acta,* 1171: 338669, doi:10.1016/j.aca.2021.338669

61. Behrmann, J., Etmann, C., Boskamp, T., Casadonte, R., Kriegsmann, J. & Maass, P., (2018), Deep Learning for Tumor Classification in Imaging Mass Spectrometry, *Bioinformatics,* 34: 1215–1223, doi:10.1093/bioinformatics/btx724

62. Seddiki, K., Saudemont, P., Precioso, F., Ogrinc, N., Wisztorski, M., Salzet, M., Fournier, I. & Droit, A., (2020), Cumulative Learning Enables Convolutional Neural Network Representations for Small Mass Spectrometry Data Classification, *Nature Communications,* 11: 5595, doi:10.1038/s41467-020-19354-z

63. Abdelmoula, W. M., Lopez, B. G.-C., Randall, E. C., Kapur, T., Sarkaria, J. N., White, F. M., Agar, J. N., Wells, W. M. & Agar, N. Y. R., (2021), Peak Learning of Mass Spectrometry Imaging Data Using Artificial Neural Networks, *Nature Communications,* 12: 5544, doi:10.1038/s41467-021-25744-8

64. Abdelmoula, W. M., Stopka, S. A., Randall, E. C., Regan, M., Agar, J. N., Sarkaria, J. N., Wells, W. M., Kapur, T. & Agar, N. Y. R., (2022), massNet: Integrated Processing and Classification of Spatially Resolved Mass Spectrometry Data Using Deep Learning for Rapid Tumor Delineation, *Bioinformatics,* btac032: 7, doi:10.1093/bioinformatics/btac032

65. Zhang, W., Claesen, M., Moerman, T., Groseclose, M. R., Waelkens, E., De Moor, B. & Verbeeck, N., (2021), Spatially Aware Clustering of Ion Images in Mass Spectrometry Imaging Data Using Deep Learning, *Analytical and Bioanalytical Chemistry,* 413: 2803–2819, doi:10.1007/s00216-021-03179-w

66. Chen, T., Kornblith, S., Norouzi, M. & Hinton, G., (2020), A Simple Framework for Contrastive Learning of Visual Representations, *Proceedings of the 37th International Conference on Machine Learning,* PMLR 119: 1597–1607

67. Maintz, J. B. A. & Viergever, M. A., (1998), A Survey of Medical Image Registration, *Medical Image Analysis,* 2: 1–36, doi:10.1016/S1361-8415(01)80026-8

68. Suganyadevi, S., Seethalakshmi, V. & Balasamy, K., (2022), A Review on Deep Learning in Medical Image Analysis, *International Journal of Multimedia Information Retrieval,* 11: 19–38, doi:10.1007/s13735-021-00218-1

69. Wells, W. M., Viola, P., Atsumi, H., Nakajima, S. & Kikinis, R., (1996), Multi-Modal Volume Registration by Maximization of Mutual Information, *Medical Image Analysis,* 1: 35–51, doi:10.1016/S1361-8415(01)80004-9

70. Maes, F., Collignon, A., Vandermeulen, D., Marchal, G. & Suetens, P., (1997), Multimodality Image Registration by Maximization of Mutual Information, *IEEE Transactions on Medical Imaging,* 16: 187–198, doi:10.1109/42.563664

71. Boveiri, H. R., Khayami, R., Javidan, R. & Mehdizadeh, A., (2020), Medical Image Registration Using Deep Neural Networks: A Comprehensive Review, *Computers & Electrical Engineering,* 87: 106767, doi:10.1016/j.compeleceng.2020.106767

72. Gibb, S. & Strimmer, K., (2012), MALDIquant: A Versatile R Package for the Analysis of Mass Spectrometry Data, *Bioinformatics,* 28: 2270–2271, doi:10.1093/bioinformatics/bts447

73. Bemis, K. D., Harry, A., Eberlin, L. S., Ferreira, C., van de Ven, S. M., Mallick, P., Stolowitz, M. & Vitek, O., (2015), *Cardinal* : An R Package for Statistical Analysis of Mass Spectrometry-Based Imaging Experiments: Fig. 1. *Bioinformatics,* 31: 2418–2420, doi:10.1093/bioinformatics/btv146

74. Röst, H. L., Sachsenberg, T., Aiche, S., Bielow, C., Weisser, H., Aicheler, F., Andreotti, S., Ehrlich, H.-C., Gutenbrunner, P., Kenar, E., *et al.,* (2016), OpenMS: A Flexible Open-Source Software Platform for Mass Spectrometry Data Analysis, *Nature Methods,* 13: 741–748, doi:10.1038/nmeth.3959

75. Veselkov, K., Sleeman, J., Claude, E., Vissers, J. P. C., Galea, D., Mroz, A., Laponogov, I., Towers, M., Tonge, R., Mirnezami, R., *et al.,* (2018), BASIS: High-performance Bioinformatics Platform for Processing of Large-Scale Mass Spectrometry Imaging Data in Chemically Augmented Histology, *Scientific Reports,* 8: 4053, doi:10.1038/s41598-018-22499-z

76. Föll, M. C., Moritz, L., Wollmann, T., Stillger, M. N., Vockert, N., Werner, M., Bronsert, P., Rohr, K., Grüning, B. A. & Schilling, O., (2019), Accessible and Reproducible Mass Spectrometry Imaging Data Analysis in Galaxy, *GigaScience,* 8: giz143, doi:10.1093/gigascience/giz143

77. *SCiLS Lab* https://www.bruker.com/ Accessed 2-2-2024

78. Ràfols, P., Torres, S., Ramírez, N., del Castillo, E., Yanes, O., Brezmes, J. & Correig, X., (2017), rMSI: An R Package for MS Imaging Data Handling and Visualization, *Bioinformatics,* 33: 2427–2428, doi:10.1093/bioinformatics/btx182

79. Ràfols, P., Heijs, B., del Castillo, E., Yanes, O., McDonnell, L. A., Brezmes, J., Pérez-Taboada, I., Vallejo, M., García-Altares, M. & Correig, X., (2020), rMSIproc: An R Package for Mass Spectrometry Imaging Data Processing, *Bioinformatics,* 36: 3618–3619, doi:10.1093/bioinformatics/btaa142

80. Robichaud, G., Garrard, K. P., Barry, J. A. & Muddiman, D. C., (2013), 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,* 24: 718–721, doi:10.1007/s13361-013-0607-z

81. Parry, R. M., Galhena, A. S., Gamage, C. M., Bennett, R. V., Wang, M. D. & Fernández, F. M., (2013), OmniSpect: An Open MATLAB-Based Tool for Visualization and Analysis of Matrix-Assisted Laser Desorption/Ionization and Desorption Electrospray Ionization Mass Spectrometry Images, *Journal of the American Society for Mass Spectrometry,* 24: 646–649, doi:10.1007/s13361-012-0572-y

82. Källback, P., Nilsson, A., Shariatgorji, M. & Andrén, P. E., (2016), msIQuant – Quantitation Software for Mass Spectrometry Imaging Enabling Fast Access, Visualization, and Analysis of Large Data Sets, *Analytical Chemistry,* 88: 4346–4353, doi:10.1021/acs.analchem.5b04603

83. He, J., Huang, L., Tian, R., Li, T., Sun, C., Song, X., Lv, Y., Luo, Z., Li, X. & Abliz, Z., (2018), MassImager: A Software for Interactive and in-Depth Analysis of Mass Spectrometry Imaging Data, *Analytica Chimica Acta,* 1015: 50–57, doi:10.1016/j.aca.2018.02.030

84. Alexandrov, T., Ovchinnikova, K., Palmer, A., Kovalev, V., Tarasov, A., Stuart, L., Nigmetzianov, R., Fay, D., Contributors, K. M., Gaudin, M., *et al. METASPACE: A Community-Populated Knowledge Base of Spatial Metabolomes in Health and Disease* Feb. 2019 Accessed 2-2-2024

85. *Lipostar* https://www.moldiscovery.com/ Accessed 11-4-2023

86. Shankar, V., Tibshirani, R. & Zare, R. N., (2021), MassExplorer: A Computational Tool for Analyzing Desorption Electrospray Ionization Mass Spectrometry Data, *Bioinformatics (Oxford, England):* btab282, doi:10.1093/bioinformatics/btab282

87. McCormick, M., Liu, X., Jomier, J., Marion, C. & Ibanez, L., (2014), ITK: Enabling Reproducible Research and Open Science, *Frontiers in Neuroinformatics,* 8, doi:10.3389/fninf.2014.00013

88. Klein, S., Staring, M., Murphy, K., Viergever, M. & Pluim, J., (2010), Elastix: A Toolbox for Intensity-Based Medical Image Registration, *IEEE Transactions on Medical Imaging,* 29: 196–205, doi:10.1109/TMI.2009.2035616

89. Schneider, C. A., Rasband, W. S. & Eliceiri, K. W., (2012), NIH Image to ImageJ: 25 Years of Image Analysis, *Nature Methods,* 9: 671–675, doi:10.1038/nmeth.2089

90. Stirling, D. R., Swain-Bowden, M. J., Lucas, A. M., Carpenter, A. E., Cimini, B. A. & Goodman, A., (2021), CellProfiler 4: Improvements in Speed, Utility and Usability, *BMC Bioinformatics,* 22: 433, doi:10.1186/s12859-021-04344-9

91. de Chaumont, F., Dallongeville, S., Chenouard, N., Hervé, N., Pop, S., Provoost, T., Meas-Yedid, V., Pankajakshan, P., Lecomte, T., Le Montagner, Y., *et al.,* (2012), Icy: An Open Bioimage Informatics Platform for Extended Reproducible Research, *Nature Methods,* 9: 690–696, doi:10.1038/nmeth.2075

92. Wolf, I., Vetter, M., Wegner, I., Böttger, T., Nolden, M., Schöbinger, M., Hastenteufel, M., Kunert, T. & Meinzer, H.-P., (2005), The Medical Imaging Interaction Toolkit, *Medical Image Analysis,* 9: 594–604, doi:10.1016/j.media.2005.04.005

93. Nolden, M., Zelzer, S., Seitel, A., Wald, D., Müller, M., Franz, A. M., Maleike, D., Fangerau, M., Baumhauer, M., Maier-Hein, L., *et al.,* (2013), The Medical Imaging Interaction Toolkit: Challenges and Advances : 10 Years of Open-Source Development, *International Journal of Computer Assisted Radiology and Surgery,* 8: 607–620, doi:10.1007/s11548-013-0840-8

94. Fedorov, A., Beichel, R., Kalpathy-Cramer, J., Finet, J., Fillion-Robin, J.-C., Pujol, S., Bauer, C., Jennings, D., Fennessy, F., Sonka, M., *et al.,* (2012), 3D Slicer as an Image Computing Platform for the Quantitative Imaging Network, *Magnetic Resonance Imaging,* 30: 1323–1341, doi:10.1016/j.mri.2012.05.001

95. Spiriev, T., Nakov, V., Laleva, L. & Tzekov, C., (2017), OsiriX Software as a Preoperative Planning Tool in Cranial Neurosurgery: A Step-by-Step Guide for Neurosurgical Residents, *Surgical Neurology International,* 8: 241, doi:10.4103/sni.sni_419_16

96. Beg, M., Taka, J., Kluyver, T., Konovalov, A., Ragan-Kelley, M., Thiery, N. M. & Fangohr, H., (2021), Using Jupyter for Reproducible Scientific Workflows, *Computing in Science & Engineering,* 23: 36–46, doi:10.1109/MCSE.2021.3052101

97. *RStudio* http://www.rstudio.com/ Accessed 3-4-2024

98. Allan, C., Burel, J.-M., Moore, J., Blackburn, C., Linkert, M., Loynton, S., Macdonald, D., Moore, W. J., Neves, C., Patterson, A., *et al.,* (2012), OMERO: Flexible, Model-Driven Data Management for Experimental Biology, *Nature Methods,* 9: 245–253, doi:10.1038/nmeth.1896

99. März, K., Franz, A. M., Seitel, A., Winterstein, A., Bendl, R., Zelzer, S., Nolden, M., Meinzer, H. .-. & Maier-Hein, L., (2014), MITK-US: Real-Time Ultrasound Support within MITK, *International Journal of Computer Assisted Radiology and Surgery,* 9: 411–420, doi:10.1007/s11548-013-0962-z

100. Nuñez, J. R., Anderton, C. R. & Renslow, R. S., (2018), Optimizing Colormaps with Consideration for Color Vision Deficiency to Enable Accurate Interpretation of Scientific Data, *PLOS ONE,* 13: e0199239, doi:10.1371/journal.pone.0199239

101. Knizner, K. T., Kibbe, R. R., Garrard, K. P., Nuñez, J. R., Anderton, C. R. & Muddiman, D. C., (2022), On the Importance of Color in Mass Spectrometry Imaging, *Journal of Mass Spectrometry,* 57: e4898, doi:10.1002/jms.4898

102. *Turbo, An Improved Rainbow Colormap for Visualization* https://blog.research.google/2019/08/turbo-improved-rainbow-colormap-for.html Accessed 11-4-2024

103. Maleike, D., Nolden, M., Meinzer, H.-P. & Wolf, I., (2009), Interactive Segmentation Framework of the Medical Imaging Interaction Toolkit, *Computer Methods and Programs in Biomedicine,* 96: 72–83, doi:10.1016/j.cmpb.2009.04.004

104. Wallner, J., Schwaiger, M., Hochegger, K., Gsaxner, C., Zemann, W. & Egger, J., (2019), A Review on Multiplatform Evaluations of Semi-Automatic Open-Source Based Image Segmentation for Cranio-Maxillofacial Surgery, *Computer Methods and Programs in Biomedicine,* 182: 105102, doi:10.1016/j.cmpb.2019.105102

105. Wasserthal, J., Breit, H.-C., Meyer, M. T., Pradella, M., Hinck, D., Sauter, A. W., Heye, T., Boll, D., Cyriac, J., Yang, S., *et al. TotalSegmentator: Robust Segmentation of 104 Anatomical Structures in CT Images* June 2023. arXiv: 2208.05868 [cs, eess] Accessed 11-7-2023

106. Isensee, F., Jaeger, P. F., Kohl, S. A. A., Petersen, J. & Maier-Hein, K. H., (2021), nnU-Net: A Self-Configuring Method for Deep Learning-Based Biomedical Image Segmentation, *Nature Methods,* 18: 203–211, doi:10.1038/s41592-020-01008-z

107. Floca, R. *MatchPoint: On Bridging the Innovation Gap between Algorithmic Research and Clinical Use in Image Registration* in *World Congress on Medical Physics and Biomedical Engineering, September 7 - 12, 2009, Munich, Germany* (eds Dössel, O. & Schlegel, W. C.) (Springer, Berlin, Heidelberg, 2010), 1105–1108

108. Schroeder, W., Martin, K. & Lorensen, B. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics* (2006)

109. *Commontk/CTK* https://github.com/commontk/CTK. Feb. 2024 Accessed 27-2-2024

110. *Qt* https://www.qt.io Accessed 11-4-2024

111. *OSGi Working Group | The Eclipse Foundation* https://www.osgi.org/ Accessed 11-4-2024

112. *Docker Engine Overview* https://docs.docker.com/engine/. May 2023 Accessed 11-4-2024

113. Menegidio, F. B., Aciole Barbosa, D., Gonçalves, R. D. S., Nishime, M. M., Jabes, D. L., Costa de Oliveira, R. & Nunes, L. R., (2019), Bioportainer Workbench: A Versatile and User-Friendly System That Integrates Implementation, Management, and Use of Bioinformatics Resources in Docker Environments, *GigaScience,* 8, doi:10.1093/gigascience/giz041

114. *PubMed Search: Keywords Docker 2010-2023* https://pubmed.ncbi.nlm.nih.gov/?term=Docker&filter=years.2010-2023&timeline=expanded&sort=pubdate Accessed 15-4-2024

115. Cordes, J., Enzlein, T., Marsching, C., Hinze, M., Engelhardt, S., Hopf, C. & Wolf, I., (2021), M^{2}aia—Interactive, Fast, and Memory-Efficient Analysis of 2D and 3D Multi-Modal Mass Spectrometry Imaging Data, *GigaScience,* 10: giab049, doi:10.1093/gigascience/giab049

116. Cordes, J., Enzlein, T., Sammour, D. A., Hopf, C. & Wolf, I., (2022), M^{2}aia Extension for Accessible Annotation Creation and Annotation Transfer for Mass Spectrometry Imaging in Multi-Modal Setups, *International Mass Spectrometry Conference:* 135–137

117. Cordes, J. & Wolf, I., (2023), MITK Docker: An Open, Language-Independent Interface for Integrating Image Processing Pipelines into MITK, *6TH Conference on Image-guided Interventions*

118. Cordes, J., Enzlein, T., Hopf, C. & Wolf, I., (2024), pyM2aia: Python Interface for Mass Spectrometry Imaging with Focus on Deep Learning, *Bioinformatics,* 40: btae133, doi:10.1093/bioinformatics/btae133

119. Fritzsche, K. H., Neher, P. F., Reicht, I., van Bruggen, T., Goch, C., Reisert, M., Nolden, M., Zelzer, S., Meinzer, H.-P. & Stieltjes, B., (2012), MITK Diffusion Imaging, *Methods of Information in Medicine,* 51: 441–448, doi:10.3414/ME11-02-0031

120. Götz, M., Nolden, M. & Maier-Hein, K., (2019), MITK Phenotyping: An Open-Source Toolchain for Image-Based Personalized Medicine with Radiomics, *Radiotherapy and Oncology*, 131: 108–111, doi:10.1016/j.radonc.2018.11.021

121. Fischer, C. R., Ruebel, O. & Bowen, B. P., (2016), An Accessible, Scalable Ecosystem for Enabling and Sharing Diverse Mass Spectrometry Imaging Analyses, *Archives of Biochemistry and Biophysics*, 589: 18–26, doi:10.1016/j.abb.2015.08.021

122. Schulz, S., Becker, M., Groseclose, M. R., Schadt, S. & Hopf, C., (2019), Advanced MALDI Mass Spectrometry Imaging in Pharmaceutical Research and Drug Development, *Current Opinion in Biotechnology*, 55: 51–59, doi:10.1016/j.copbio.2018.08.003

123. Meuleman, W., Engwegen, J. Y., Gast, M.-C. W., Beijnen, J. H., Reinders, M. J. & Wessels, L. F., (2008), Comparison of Normalisation Methods for Surface-Enhanced Laser Desorption and Ionisation (SELDI) Time-of-Flight (TOF) Mass Spectrometry Data, *BMC Bioinformatics*, 9: 88, doi:10.1186/1471-2105-9-88

124. Shin, H. & Markey, M. K., (2006), A Machine Learning Perspective on the Development of Clinical Decision Support Systems Utilizing Mass Spectra of Blood Samples, *Journal of Biomedical Informatics*, 39: 227–248, doi:10.1016/j.jbi.2005.04.002

125. Williams, B., Cornett, S., Dawant, B., Crecelius, A., Bodenheimer, B. & Caprioli, R. *An Algorithm for Baseline Correction of MALDI Mass Spectra* in *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 1* (ACM, Kennesaw Georgia, Mar. 2005), 137–142 Accessed 4-4-2024

126. Liu, L.-H., Shan, B.-E., Tian, Z.-Q., Sang, M.-X., Ai, J., Zhang, Z.-F., Meng, J., Zhu, H. & Wang, S.-J., (2010), Potential Biomarkers for Esophageal Carcinoma Detected by Matrix-Assisted Laser Desorption/Ionization Time-of-Flight Mass Spectrometry, *Clinical Chemistry and Laboratory Medicine*, 48: 855–861, doi:10.1515/CCLM.2010.138

127. van Herk, M., (1992), A Fast Algorithm for Local Minimum and Maximum Filters on Rectangular and Octagonal Kernels, *Pattern Recognition Letters*, 13: 517–521, doi:10.1016/0167-8655(92)90069-C

128. Cordes, J., Enzlein, T., Marsching, C., Marven, H., Engelhardt, S., Hopf, C. & Wolf, I., (2021), Supporting Data for "M^{2}aia - Interactive, Fast and Memory Efficient Analysis of 2D and 3D Multi-Modal Mass Spectrometry Imaging Data", doi:10.5524/100909

129. Tibshirani, R., Hastie, T., Narasimhan, B., Soltys, S., Shi, G., Koong, A. & Le, Q.-T., (2004), Sample Classification from Protein Mass Spectrometry, by 'Peak Probability Contrasts', *Bioinformatics*, 20: 3034–3044, doi:10.1093/bioinformatics/bth357

130. Coombes, K. R., Tsavachidis, S., Morris, J. S., Baggerly, K. A., Hung, M.-C. & Kuerer, H. M., (2005), Improved Peak Detection and Quantification of Mass Spectrometry Data Acquired from Surface-Enhanced Laser Desorption and Ionization by Denoising Spectra with the Undecimated Discrete Wavelet Transform, *PROTEOMICS*, 5: 4107–4117, doi:10.1002/pmic.200401261

131. Purohit, P. V. & Rocke, D. M., (2003), Discriminant Models for High-throughput Proteomics Mass Spectrometer Data, *PROTEOMICS*, 3: 1699–1703, doi:10.1002/pmic.200300518

132. Yang, H.-c., Dasdan, A., Hsiao, R.-L. & Parker, D. S. *Map-Reduce-Merge: Simplified Relational Data Processing on Large Clusters* in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data* (Association for Computing Machinery, New York, NY, USA, June 2007), 1029–1040 Accessed 28-3-2024

133. Breen, E., Hopewood, F., Williams, K. & Wilkins, M., (2000), Automatic Poisson Peak Harvesting for High Throughput Protein Identification, *ELECTROPHORESIS ER,* 21: 2243–2251, doi:10.1002/1522-2683(20000601)21:11<2243::AID-ELPS2243>3.0.CO;2-K

134. Bro, R. & Smilde, A. K., (2014), Principal Component Analysis, *Analytical Methods,* 6: 2812–2831, doi:10.1039/C3AY41907J

135. Hinton, G. & Roweis, S., (2003), Stochastic Neighbor Embedding, *Advances in Neural Information Processing Systems,* 15: 857–864

136. Schüffler, P. J., Ozcan, G. G., Al-Ahmadie, H. & Fuchs, T. J., (2021), FlexTileSource: An OpenSeadragon Extension for Efficient Whole-Slide Image Visualization, *Journal of Pathology Informatics,* 12: 31, doi:10.4103/jpi.jpi_13_21

137. Goode, A., Gilbert, B., Harkes, J., Jukic, D. & Satyanarayanan, M., (2013), OpenSlide: A Vendor-Neutral Software Foundation for Digital Pathology, *Journal of Pathology Informatics,* 4: 27, doi:10.4103/2153-3539.119005

138. *Visualization Toolkit (VTK)* http://www.vtk.org Accessed 3-4-2024

139. *VTK/Common/Core/vtkScalarsToColors.h* https://github.com/Kitware/VTK/blob/master/Common/Core/vtkScalarsToColors.h Accessed 5-5-2024

140. Fitzpatrick, J., West, J. & Maurer, C., (1998), Predicting Error in Rigid-Body Point-Based Registration, *IEEE Transactions on Medical Imaging,* 17: 694–702, doi:10.1109/42.736021

141. Li, D., Qian, Y., Yao, H., Yu, W. & Ma, X., (2023), DeepS: Accelerating 3D Mass Spectrometry Imaging via a Deep Neural Network, *Analytical Chemistry,* doi:10.1021/acs.analchem.2c05785

142. *imzML* https://imzml.github.io/ Accessed 11-4-2023

143. Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., *et al.,* (2016), The FAIR Guiding Principles for Scientific Data Management and Stewardship, *Scientific Data,* 3: 160018, doi:10.1038/sdata.2016.18

144. Thomas, S. A., Race, A. M., Steven, R. T., Gilmore, I. S. & Bunch, J., (2016), Dimensionality Reduction of Mass Spectrometry Imaging Data Using Autoencoders: 1–7, doi:10.1109/SSCI.2016.7849863

145. *Project Jupyter* https://jupyter.org Accessed 11-4-2023

146. *Visual Studio Code - Code Editing* https://code.visualstudio.com Accessed 11-4-2024

147. *Dorowu/Ubuntu-Desktop-Lxde-Vnc - Docker Image | Docker Hub* https://hub.docker.com/r/dorowu/ubuntu-desktop-lxde-vnc Accessed 11-4-2023

148. *Desktop › Wiki › Ubuntuusers.De* https://wiki.ubuntuusers.de/Desktop/ Accessed 11-4-2023

149. *Fcwu/Docker-Ubuntu-Vnc-Desktop: A Docker Image to Provide Web VNC Interface to Access Ubuntu LXDE/LxQT Desktop Environment.* https://github.com/fcwu/docker-ubuntu-vnc-desktop Accessed 11-4-2023

150. Abu Sammour, D., Cairns, J. L., Boskamp, T., Marsching, C., Kessler, T., Ramallo Guevara, C., Panitz, V., Sadik, A., Cordes, J., Schmidt, S., *et al.,* (2023), Spatial Probabilistic Mapping of Metabolite Ensembles in Mass Spectrometry Imaging, *Nature Communications,* 14: 1823, doi:10.1038/s41467-023-37394-z

151. Gustafsson, O. J. R., Briggs, M. T., Condina, M. R., Winderbaum, L. J., Pelzing, M., McColl, S. R., Everest-Dass, A. V., Packer, N. H. & Hoffmann, P., (2015), MALDI Imaging Mass Spectrometry of N-linked Glycans on Formalin-Fixed Paraffin-Embedded Murine Kidney, *Analytical and Bioanalytical Chemistry,* 407: 2127–2139, doi:10.1007/s00216-014-8293-7

152. Gustafsson, O. J., Briggs, M. T., Condina, M. R., Winderbaum, L. J., Pelzing, M., McColl, S. R., Everest-Dass, A. V., Packer, N. H. & Hoffmann, P., (2018), Raw N-glycan Mass Spectrometry Imaging Data on Formalin-Fixed Mouse Kidney, *Data in Brief,* 21: 185–188, doi:10.1016/j.dib.2018.08.186

153. Tibshirani, R., Hastie, T., Narasimhan, B. & Chu, G., (2003), Class Prediction by Nearest Shrunken Centroids, with Applications to DNA Microarrays, *Statistical Science,* 18: 104–117, doi:10.1214/ss/1056397488

154. Bemis, K. D., Harry, A., Eberlin, L. S., Ferreira, C. R., van de Ven, S. M., Mallick, P., Stolowitz, M. & Vitek, O., (2016), Probabilistic Segmentation of Mass Spectrometry (MS) Images Helps Select Important Ions and Characterize Confidence in the Resulting Segments, *Molecular & Cellular Proteomics,* 15: 1761–1772, doi:10.1074/mcp.O115.053918

155. Cordes, J., (2021), Supporting protocol for use-case 1: N-linked glycan m/z candidate detection in "M2aia - Interactive, fast and memory efficient analysis of 2D and 3D multi-modal mass spectrometry imaging data", *protocols.io,* doi:10.17504/protocols.io.bvq5n5y6

156. Cordes, J., (2021), Supporting protocol for use-case 1: Dimensionality reduction in "M2aia - Interactive, fast and memory efficient analysis of 2D and 3D multi-modal mass spectrometry imaging data", *protocols.io,* doi:10.17504/protocols.io.brw4m7gw

157. Cordes, J., (2021), Supporting capsule for use-case 1: R-based processing in "M²aia - Interactive, fast and memory efficient analysis of 2D and 3D multi-modal mass spectrometry imaging data", *CodeOcean,* doi:10.24433/CO.2384502.v1

158. *M²aia Github repository; R-based processing for use-case 1 "N-linked glycan m/z candidate detection"* https://github.com/jtfcordes/m2aia/blob/v2021.01.01/Scripts/N-Glycan-CandidateDetection_UseCase1.R Accessed 3-4-2024

159. Cordes, J., (2021), Supporting capsule for use-case 1: Command-line application based pre-processing in "M²aia - Interactive, fast and memory efficient analysis of 2D and 3D multi-modal mass spectrometry imaging data", *CodeOcean,* doi:10.24433/CO.7662658.v1

160. *M²aia Github repository; Command-line application for use-case 1 "N-linked glycan m/z candidate detection"* https://github.com/jtfcordes/m2aia/blob/v2021.01.01/Modules/M2aiaCLI/NLinkedGlycan.cpp Accessed 3-4-2024

161. Cordes, J., (2021), Supporting protocol for use-case 2: Multi-modal 3D image reconstruction in "M2aia - Interactive, fast and memory efficient analysis of 2D and 3D multi-modal mass spectrometry imaging data", *protocols.io,* doi:10.17504/protocols.io.bvq8n5zw

162. *M2aia/Pym2aia-Examples: Memory-Efficient Python Interface for Mass Spectrometry Imaging with Focus on Deep Learning* https://github.com/m2aia/pym2aia-examples Accessed 1-5-2024

163. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems* Mar. 2016. arXiv: 1603.04467 [cs] Accessed 27-2-2024

164. *The HDF5® Library & File Format* https://www.hdfgroup.org/ Accessed 8-4-2024

165. *M2aia/Pym2aia-Examples-msiPL* https://github.com/m2aia/pym2aia-examples-msiPL. Oct. 2022 Accessed 4-5-2024

166. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., *et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library* Dec. 2019. arXiv: 1912.01703 [cs, stat] Accessed 27-2-2024

167. Tan, M. & Le, Q. V. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* Sept. 2020. arXiv: 1905.11946 [cs, stat] Accessed 2-9-2022

168. McInnes, L., Healy, J. & Melville, J. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction* Sept. 2020. arXiv: 1802.03426 [cs, stat] Accessed 17-5-2022

169. Damle, A., Minden, V. & Ying, L., (2019), Simple, Direct and Efficient Multi-Way Spectral Clustering, *Information and Inference: A Journal of the IMA,* 8: 181–203, doi:10.1093/imaiai/iay008

170. Smets, T., Waelkens, E. & De Moor, B., (2020), Prioritization of m / z -Values in Mass Spectrometry Imaging Profiles Obtained Using Uniform Manifold Approximation and Projection for Dimensionality Reduction, *Analytical Chemistry,* 92: 5240–5248, doi:10.1021/acs.analchem.9b05764

171. Gardner, W., Cutts, S. M., Phillips, D. R. & Pigram, P. J., (2021), Understanding Mass Spectrometry Images: Complexity to Clarity with Machine Learning, *Biopolymers,* 112, doi:10.1002/bip.23400

172. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction — Umap 0.5 Documentation* https://umap-learn.readthedocs.io/en/latest/ Accessed 5-5-2024

173. *Wasserth/Totalsegmentator Tags | Docker Hub* https://hub.docker.com/r/wasserth/totalsegmentator/tags Accessed 4-5-2024

174. Wasserthal, J. *Wasserth/TotalSegmentator* https://github.com/wasserth/TotalSegmentator. May 2024 Accessed 4-5-2024

175. *MITK-Data · Mitkdata* https://phabricator.mitk.org/source/mitkdata/ Accessed 4-5-2024

176. *M2aia/Mitk-Docker: This Project Provides Utilities to Run Docker Commands from within MITK and Docker Based Processing Views.* https://github.com/m2aia/mitk-docker Accessed 5-5-2024

177. *M2aia/M2aia* https://github.com/m2aia/m2aia. Apr. 2024 Accessed 5-5-2024

178. Bankhead, P., Loughrey, M. B., Fernández, J. A., Dombrowski, Y., McArt, D. G., Dunne, P. D., McQuaid, S., Gray, R. T., Murray, L. J., Coleman, H. G., *et al.,* (2017), QuPath: Open Source Software for Digital Pathology Image Analysis, *Scientific Reports,* 7: 16878, doi:10.1038/s41598-017-17204-5

179. Berg, S., Kutra, D., Kroeger, T., Straehle, C. N., Kausler, B. X., Haubold, C., Schiegg, M., Ales, J., Beier, T., Rudy, M., *et al.,* (2019), Ilastik: Interactive Machine Learning for (Bio)Image Analysis, *Nature Methods,* 16: 1226–1232, doi:10.1038/s41592-019-0582-9

180. Iakab, S.-A., Keller, F., Schmidt, S., Cordes, J., Zhou, Q., Cairns, J. L., Fischer, F., Schneider, R., Wolf, I., Rudolf, R., *et al. 3D-Mass Spectrometry Imaging of Micro-scale 3D Cell Culture Models in Cancer Research* Dec. 2022 Accessed 3-4-2024

181. *Teem: Nrrd: Definition of NRRD File Format* https://teem.sourceforge.net/nrrd/format.html Accessed 18-5-2024

182. *Supporting Material for "3D-Mass Spectrometry Imaging of Micro-scale 3D Cell Culture Models in Cancer Research" | bioRxiv* https://www.biorxiv.org/content/10.1101/2022.12.05.519157v1.supplementary-material Accessed 25-5-2024

183. Razeghi, O., Solís-Lemus, J. A., Lee, A. W., Karim, R., Corrado, C., Roney, C. H., de Vecchi, A. & Niederer, S. A., (2020), CemrgApp: An Interactive Medical Imaging Application with Image Processing, Computer Vision, and Machine Learning Toolkits for Cardiovascular Research, *SoftwareX,* 12: 100570, doi:10.1016/j.softx.2020.100570

Publications

Cordes, J.; Enzlein, T.; Marsching, C.; Hinze, M.; Engelhardt, S.; Hopf, C.; Wolf, I. M²aia—Interactive, Fast, and Memory-Efficient Analysis of 2D and 3D Multi-Modal Mass Spectrometry Imaging Data. GigaScience 2021, 10 (7)

Cordes, J.; Enzlein, T.; Hopf, C.; Wolf, I. pyM2aia: Python Interface for Mass Spectrometry Imaging with Focus on Deep Learning. Bioinformatics 2024, btae133

Cordes, J.; Enzlein, T.; Sammour, D. A.; Hopf, C.; Wolf, I. M²aia Extension for Accessible Annotation Creation and Annotation Transfer for Mass Spectrometry Imaging in Multi-Modal Setups. In International Mass Spectrometry Conference; Maastricht, 2022; Vol. Abstract ID 391, pp 135–137.

Cordes, J.; Wolf, I. MITK Docker: An Open, Language-Independent Interface for Integrating Image Processing Pipelines into MITK. In 6TH Conference on Image-guided Interventions; Mannheim, 2023.

Enzlein, T.; **Cordes, J.**; Munteanu, B.; Michno, W.; Serneels, L.; De Strooper, B.; Hanrieder, J.; Wolf, I.; Chávez-Gutiérrez, L.; Hopf, C. Computational Analysis of Alzheimer Amyloid Plaque Composition in 2D- and Elastically Reconstructed 3D-MALDI MS Images. Anal. Chem. 2020, acs.analchem.0c02585

Abu Sammour, D.; Cairns, J. L.; Boskamp, T.; Marsching, C.; Kessler, T.; Ramallo Guevara, C.; Panitz, V.; Sadik, A.; **Cordes, J.**; Schmidt, S.; Mohammed, S. A.; Rittel, M. F.; Friedrich, M.; Platten, M.; Wolf, I.; von Deimling, A.; Opitz, C. A.; Wick, W.; Hopf, C. Spatial Probabilistic Mapping of Metabolite Ensembles in Mass Spectrometry Imaging. Nat Commun 2023, 14 (1), 1823.

Iakab, S.-A.; Keller, F.; Schmidt, S.; **Cordes, J.**; Zhou, Q.; Cairns, J. L.; Fischer, F.; Schneider, R.; Wolf, I.; Rudolf, R.; Hopf, C. 3D-Mass Spectrometry Imaging of Micro-Scale 3D Cell Culture Models in Cancer Research. bioRxiv December 8, 2022, p 2022.12.05.519157.

Curriculum Vitae

Personal Data

Name:	Jonas Tom Frederik Cordes
Place and Date of Birth:	Heidelberg, Germany | 15 February 1990
Family Status:	Married, two children

Education

11/2018-Present	Doctoral Candidate Medical Faculty Mannheim, Heidelberg University, Mannheim, Germany Supervised by Prof. Dr. Ivo Wolf at the Faculty of Computer Science Hochschule Mannheim, Mannheim, Germany
09/2012-02/2017	Master of Science in Applied Computer Science Faculty of Mathematics and Computer Science Ruprecht-Karls University, Heidelberg, Germany Accumulated Grade: 1.7
09/2008-09/2012	Bachelor of Science in Medical Computer Science Faculty of Mathematics and Computer Science Hochschule Mannheim, Mannheim, Germany Accumulated Grade: 1.4
07/2008	Entrance Qualification for Universities of Applied Sciences Vocational College for Technical Communication Hans-Freudenberg-Schule, Weinheim, Germany

Work Experience

07/2017-Present	Project team member - Multimodale Analytik für die Life Science Industrie (M²Aind) and - Mannheim Molecular Intervention Environment (M²OLIE) Faculty of Computer Science, Hochschule Mannheim, Mannheim, Germany
08/2010-07/2016	Student Research Assistant Medical and biological Informatics German Cancer Research Center (DKFZ), Heidelberg, Germany

Danksagung

Diese Arbeit wurde am Institut für Medizinische Informatik der Hochschule Mannheim in Zusammenarbeit mit der der Medizinischen Fakultät Mannheim der Universität Heidelberg verfasst. Die Dissertation fasst die Arbeit zu dem Thema *A Software Ecosystem for Remote Analysis of Mass Spectrometry Imaging* zusammen. Danken möchte ich insbesondere Ivo Wolf für seine langjährige Unterstützung. Bedanken möchte ich mich auch bei meinen Kollegen Sandy Engelhardt, Thomas Enzlein und Denis Abu Sammour sowie bei den von mir betreuten Master- und Bachelorstudenten für die intensive Zusammenarbeit. Auch an Carsten Hopf den Leiter des Center of Mass Spectrometry and Optical Spectroscopy (CeMOS) geht ein Dankeschön, ebenso wie an die langjährigen Kollegen der Projektgruppen M2Aind und M2OLI.

Ich bin meiner Familie und meinen Freunden sehr dankbar für ihre Unterstützung während meines gesamten Lebens. An erster Stelle möchte ich meinen Eltern und meinen Geschwistern danken, die mir den Rücken stärken, egal was ich tue. Ein ganz besonderes Dankeschön geht an meine wundervollen Töchter Mira und Jara. Ich hab' euch lieb! Und zu guter Letzt und am allerwichtigsten: Ich möchte mich bei meiner wunderbaren Frau Nina bedanken, die mich in jeder Hinsicht unterstützt und mir immer den Rücken freihält.