Project Management and Software Development for Medical Applications

# Image Processing for Digital Breast Tomosynthesis

Intermediate Presentation, 20.06.2023

Project Supervisor: Mickael Tardy
Presented by: Berfin Kavsut

Technische Universität München
Johns Hopkins Whiting School of Engineering

Section 01

# Background

# Medical Imaging Modality

## Digital Breast Tomosynthesis [1]

- 3D images
- Reconstructed from 2D X-ray projections
- Cross-sectional visualization via slices

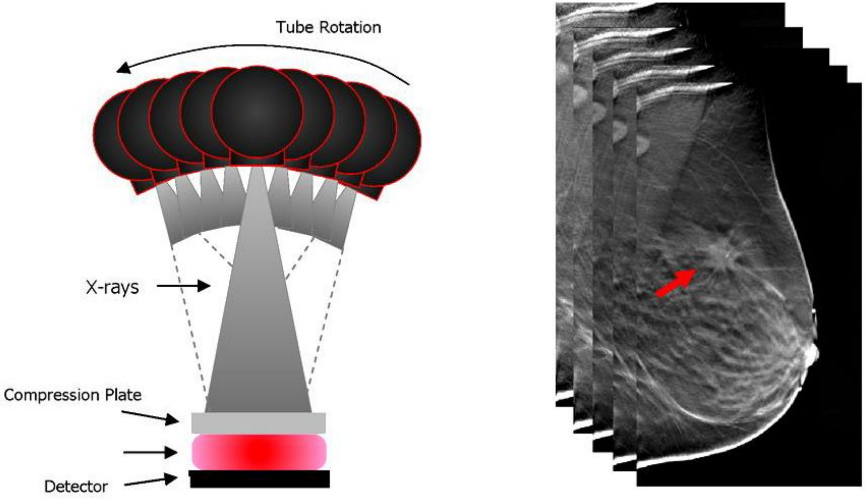



Figure 1. Acquisition Geometry of Digital Breast Tomosynthesis [2]


[1] "Breast Imaging Diagnosis and Screeningin Korea," *Breast Imaging, 12th International Workshop IWDM*, Gifu City, Japan, 2014, pp. xxvii-xxviii, doi: 10.1007/978-3-319-07887-8.

[2] D. Kontos, P. R. Bakic, and A. D. A. Maidment, "Texture in digital breast tomosynthesis: a comparison between mammographic and n tomographic characterization of parenchymal properties," in Proc. SPIE, vol. 6915, Mar. 17, 2008, p. 69150A. doi: 10.1117/12.773144.

# Slabbing

Slices:

- 1 mm thickness for standard protocol
- very high image noise level [3]

Slabs:

- 10 mm thickness for standard protocol
- better image quality [3]

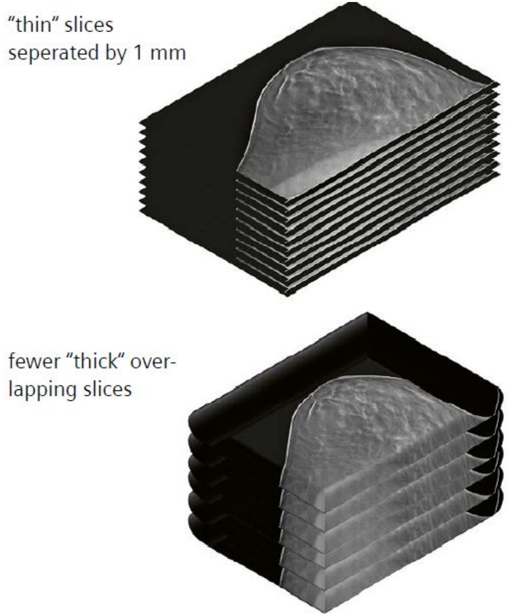



Figure 2. Slabbing slices [4]

[3] F. Diekmann, H. Meyer, S. Diekmann et al., "Thick Slices from Tomosynthesis Data Sets: Phantom Study for the Evaluation of Different Algorithms," *Journal of Digital Imaging*, vol. 22, no. 5, pp. 519-526, Oct. 2009, doi: 10.1007/s10278-007-9075-y.
[4] HealthManagement.org, "Digital Breast Tomosynthesis in screening – approaches to reduce reading time," HealthManagement, https://healthmanagement.org/c/decision-support/whitepaper/digital-breast-tomosynthesis-in-screening-approaches-to-reduce-reading-time (accessed May 13, 2023).

Section 02

# Methods

Technische Universität München + Johns Hopkins Whiting School of Engineering

# Pipeline

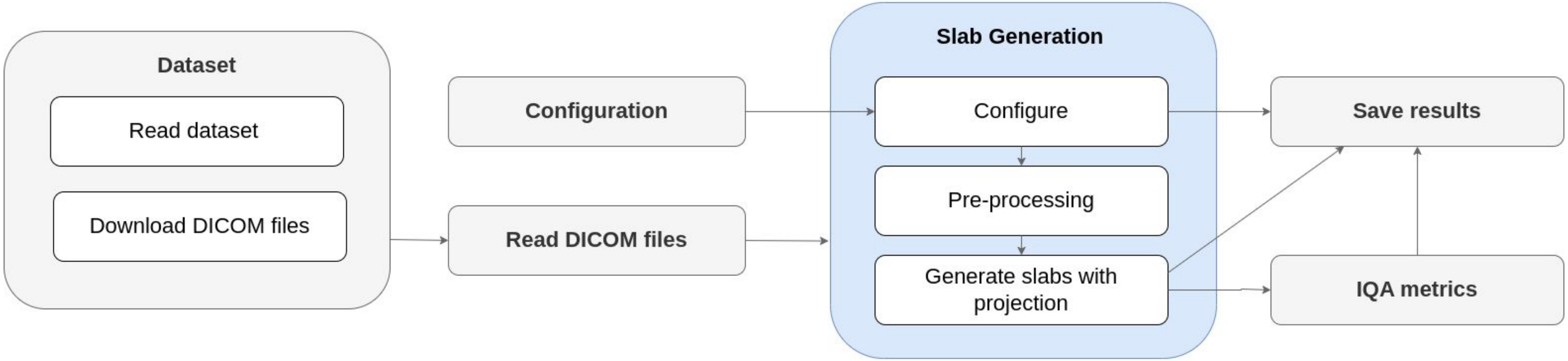



Figure 3. Pipeline schematic of the project

# Pre-Processing

- Background noise removal
- Clearing the bright breast skin

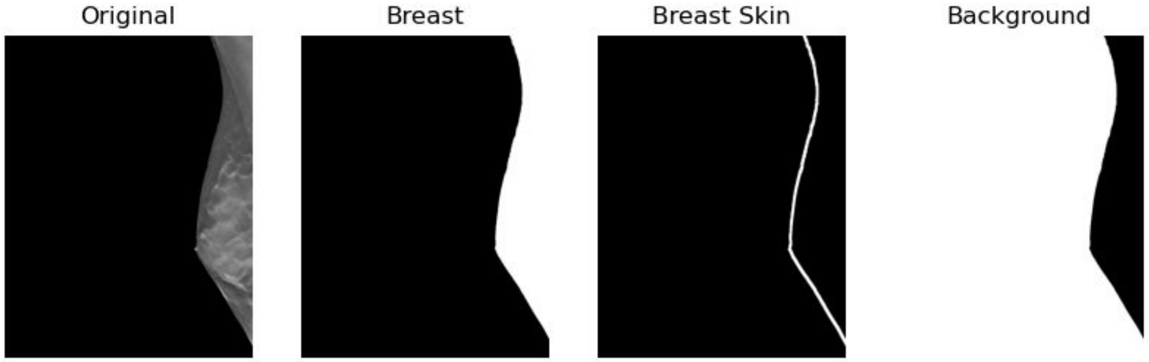



Figure 4. Breast skin segmentation

# Pre-Processing

## Breast segmentation

Steps:

1. Breast segmentation with **triangle thresholding**
2. **Lower resolution** of the segmented breast mask
3. Segment breast skin by using the **morphological operation** called erosion
4. Create breast, skin, and background masks

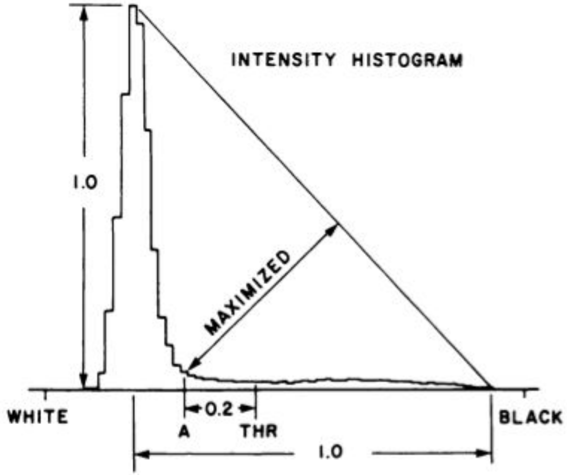



Figure 5. Triangle Thresholding [5]

[5] “Image thresholding,” OpenCV, https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html (accessed Jun. 19, 2023).

# Projection Methods

- Maximum intensity projection: $f_w^{MIP}(x) = 1 \; if \; x = 1, \; else \; 0$

- Average intensity projection: $f_w^{AIP}(x) = 1$

- SoftMIP: $f_w^{softMIP}(x) = x^4$

create projection line: $$P_s(x)\,,\; 0 \leqslant x \leqslant l$$ $$P_s\,(0) = min(P),\; P_s(l) = max(P).$$

compute projection: $$p = \frac{1}{\int_0^1 f_w(x)\,dx} \int_0^1 f_w(x)\,P_s(x)\,dx$$

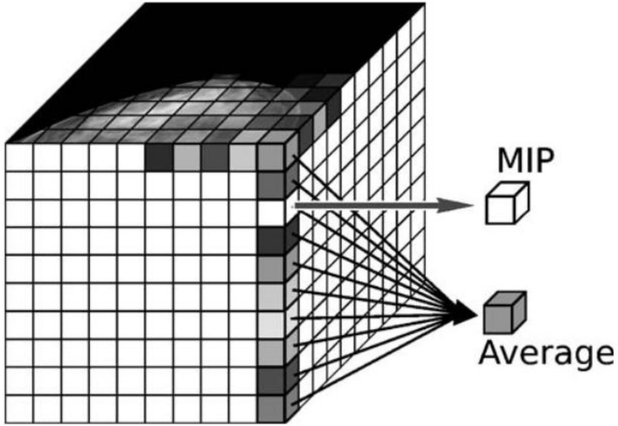



Figure 6. Maximum intensity projection and average intensity projection [6]

[6] F. Diekmann, H. Meyer, S. Diekmann et al., "Thick Slices from Tomosynthesis Data Sets: Phantom Study for the Evaluation of Different Algorithms," *Journal of Digital Imaging*, vol. 22, no. 5, pp. 519-526, Oct. 2009, doi: 10.1007/s10278-007-9075-y.

Section 03

# Software Development

# Software Tools

**Version control:** GitHub

**IDE:** PyCharm

**Issue Tracking System:** GitHub issues

**Python Libraries:**

- scikit-image
- OpenCV
- pandas
- pydicom
- numpy

# Issue Tracking

**To do | 7**

- Keep track of DICOM's SeriesInstanceUID when saving slabs
- Save config as json alongside with generated slabs
- Provide tools allowing for processing a DICOM DBT volume as input and saving slabs as output
- Choose the configuration method to call the slab generator
- Displaying pathology better (related to normalization methods)
- Normalization when creating the slabs
- ...

**In progress | 4**

- Do not apply averaging on the skin border pixel when skin-border is activated
- Add configuration parameter to slab generator to skip slices near to ends
- Provide default config values to avoid failures
- Challenge breast skin effect by choosing dicom files with thick and bright breast skins

**Done | 6**

- Read box coordinates of pathologies (20 samples)
- Use lower resolution images for breast segmentation
- Add SoftMIP projection
- Saved 10-bit images with 16-bits range
- Allow DICOMFileProcessor to use images from local storage
- Choose the morphological structure

Section 04

# Challenges

# Challenges

Design the end-to-end architecture

Run code only once

- Prepare dataset by downloading DICOM files
- Generate multiple configurations
- Save the results with configuration

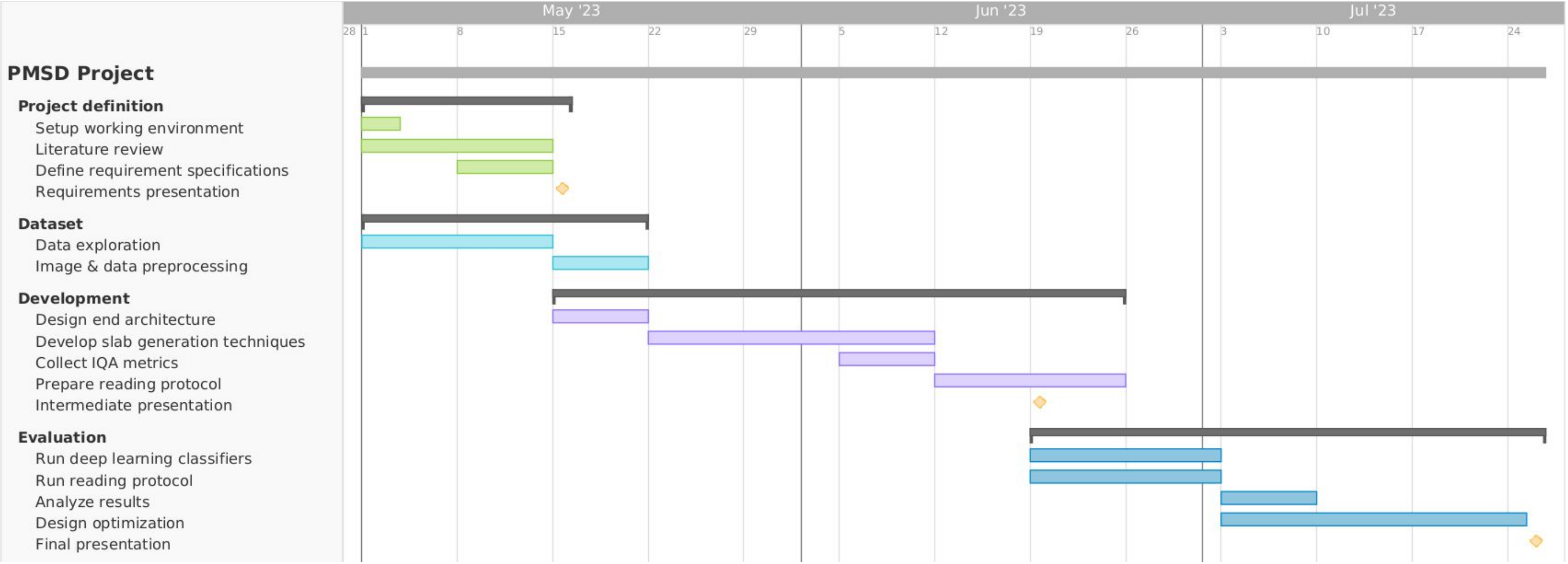
Projection with and w/o breast skin

- Keep the breast skin from center image
- Modify projection accordingly

Section 05

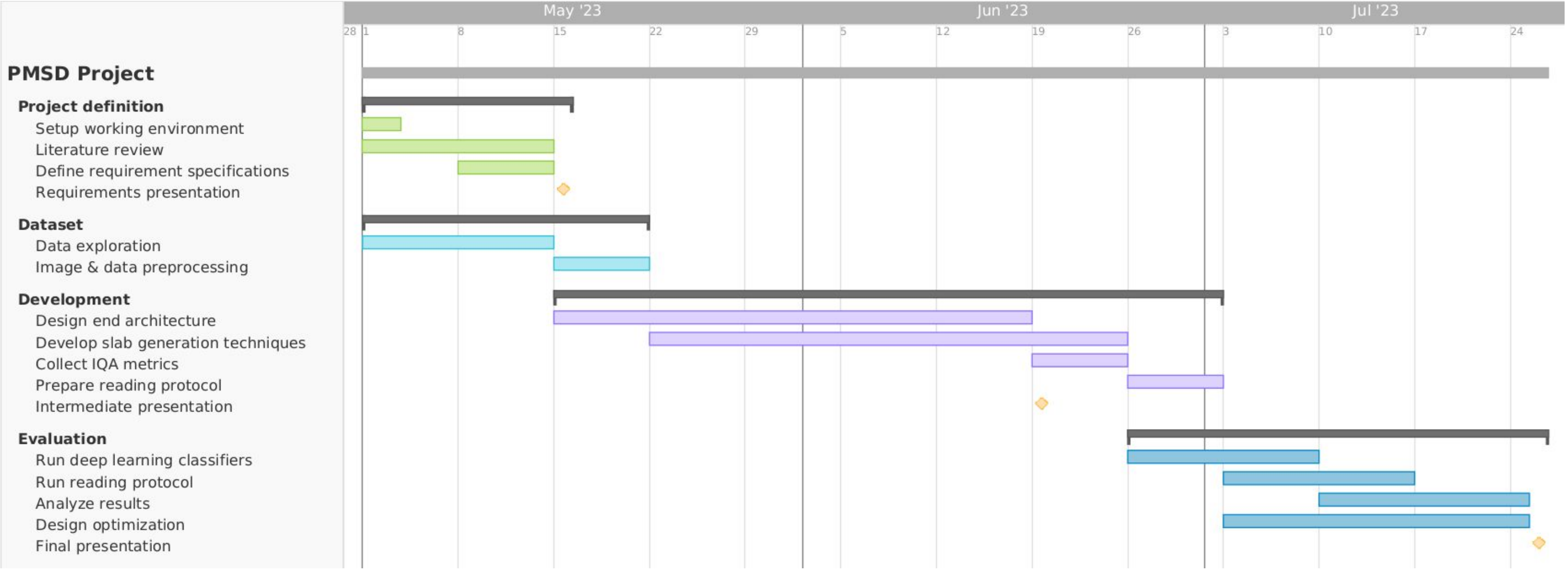
# Gantt Chart

# Gantt Chart



May '23
Jun '23
Jul '23
28
1
8
15
22
29
5
12
19
26
3
10
17
24
PMSD Project
Project definition
Setup working environment
Literature review
Define requirement specifications
Requirements presentation
Dataset
Data exploration
Image & data preprocessing
Development
Design end architecture
Develop slab generation techniques
Collect IQA metrics
Prepare reading protocol
Intermediate presentation
Evaluation
Run deep learning classifiers
Run reading protocol
Analyze results
Design optimization
Final presentation

# Updated Gantt Chart

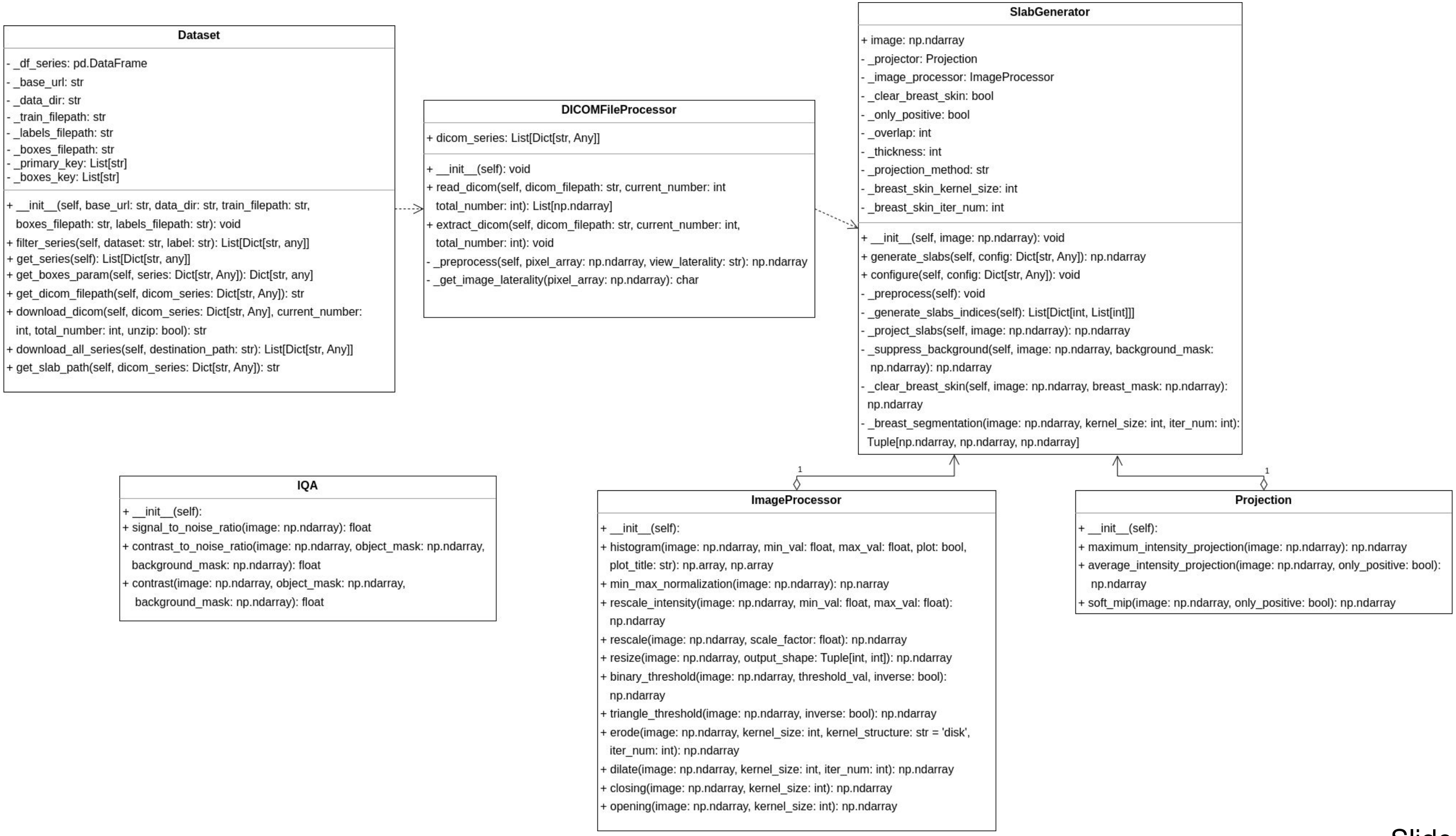


Section 06

# Appendix

# UML Class Diagram

## - Current design


Dataset
- _df_series: pd.DataFrame
- _base_url: str
- _data_dir: str
- _train_filepath: str
- _labels_filepath: str
- _boxes_filepath: str
- _primary_key: List[str]
- _boxes_key: List[str]
+ __init__(self, base_url: str, data_dir: str, train_filepath: str, boxes_filepath: str, labels_filepath: str): void
+ filter_series(self, dataset: str, label: str): List[Dict[str, any]]
+ get_series(self): List[Dict[str, any]]
+ get_boxes_param(self, series: Dict[str, Any]): Dict[str, any]
+ get_dicom_filepath(self, dicom_series: Dict[str, Any]): str
+ download_dicom(self, dicom_series: Dict[str, Any], current_number: int, total_number: int, unzip: bool): str
+ download_all_series(self, destination_path: str): List[Dict[str, Any]]
+ get_slab_path(self, dicom_series: Dict[str, Any]): str
DICOMFileProcessor
+ dicom_series: List[Dict[str, Any]]
+ __init__(self): void
+ read_dicom(self, dicom_filepath: str, current_number: int total_number: int): List[np.ndarray]
+ extract_dicom(self, dicom_filepath: str, current_number: int, total_number: int): void
- _preprocess(self, pixel_array: np.ndarray, view_laterality: str): np.ndarray
- _get_image_laterality(pixel_array: np.ndarray): char
SlabGenerator
+ image: np.ndarray
- _projector: Projection
- _image_processor: ImageProcessor
- _clear_breast_skin: bool
- _only_positive: bool
- _overlap: int
- _thickness: int
- _projection_method: str
- _breast_skin_kernel_size: int
- _breast_skin_iter_num: int
+ __init__(self, image: np.ndarray): void
+ generate_slabs(self, config: Dict[str, Any]): np.ndarray
+ configure(self, config: Dict[str, Any]): void
- _preprocess(self): void
- _generate_slabs_indices(self): List[Dict[int, List[int]]]
- _project_slabs(self, image: np.ndarray): np.ndarray
- _suppress_background(self, image: np.ndarray, background_mask: np.ndarray): np.ndarray
- _clear_breast_skin(self, image: np.ndarray, breast_mask: np.ndarray): np.ndarray
- _breast_segmentation(image: np.ndarray, kernel_size: int, iter_num: int): Tuple[np.ndarray, np.ndarray, np.ndarray]
IQA
+ __init__(self):
+ signal_to_noise_ratio(image: np.ndarray): float
+ contrast_to_noise_ratio(image: np.ndarray, object_mask: np.ndarray, background_mask: np.ndarray): float
+ contrast(image: np.ndarray, object_mask: np.ndarray, background_mask: np.ndarray): float
1
ImageProcessor
+ __init__(self):
+ histogram(image: np.ndarray, min_val: float, max_val: float, plot: bool, plot_title: str): np.array, np.array
+ min_max_normalization(image: np.ndarray): np.narray
+ rescale_intensity(image: np.ndarray, min_val: float, max_val: float): np.ndarray
+ rescale(image: np.ndarray, scale_factor: float): np.ndarray
+ resize(image: np.ndarray, output_shape: Tuple[int, int]): np.ndarray
+ binary_threshold(image: np.ndarray, threshold_val, inverse: bool): np.ndarray
+ triangle_threshold(image: np.ndarray, inverse: bool): np.ndarray
+ erode(image: np.ndarray, kernel_size: int, kernel_structure: str = 'disk', iter_num: int): np.ndarray
+ dilate(image: np.ndarray, kernel_size: int, iter_num: int): np.ndarray
+ closing(image: np.ndarray, kernel_size: int): np.ndarray
+ opening(image: np.ndarray, kernel_size: int): np.ndarray
1
Projection
+ __init__(self):
+ maximum_intensity_projection(image: np.ndarray): np.ndarray
+ average_intensity_projection(image: np.ndarray, only_positive: bool): np.ndarray
+ soft_mip(image: np.ndarray, only_positive: bool): np.ndarray