Module Interface Specification for FBP CT Image Reconstruction

Qianlin Chen

March 18, 2025

1 Revision History

Date	Version	Notes
March 11	1.0	Initial document

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [give url —SS]

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

3 Introduction

The following document details the Module Interface Specifications for FBP CT Image Reconstruction. Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at MIS.

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by FBP CT Image Reconstruction.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of FBP CT Image Reconstruction uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, FBP CT Image Reconstruction uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding	Hardware-Hiding Module
	Filter Module
	FBP Module
	IO Handler Module
Behaviour-Hiding	Services Module
Software Decision	Sinogram Simulation Module

Table 1: Module Hierarchy

6 MIS of Filter Module

6.1 Module

Filter

6.2 Uses

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_fourier_filter	s: int, filter_name: str	filter: $\mathbb{R}^s$	OutOfRange, NULL

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Environment Variables

None

6.4.3 Assumptions

Input size s is a positive number.

6.4.4 Access Routine Semantics

get_filter(s, filter_name):

- output: out := $2\Re(\mathcal{F}(f(s))$ with
  $f(s) := (filter\_name =' shepp' \Rightarrow f(s) \mid\mid filter\_name =' ramp' \Rightarrow g(s) \mid\mid filter\_name = None \Rightarrow None)$ where:

  –

$$f(s) = \begin{cases} 0.25, & s = 0 \\ \dfrac{-1}{\pi s^2}, & \text{s is odd} \\ 0, & \text{s is even} \end{cases} \tag{1}$$

–

$$g(s) = \frac{\sin \pi \cdot freq(n)}{\pi \cdot freq(n)} \cdot freq(n) \tag{2}$$

freq(n) corresponds to the frequency bins in the discrete Fourier domain:

$$freq(n) = \begin{cases} \frac{n}{s}, & 0 \leq n < \frac{s}{2} \\ \frac{n-s}{s}, & \frac{s}{2} \leq n < s \end{cases} \tag{3}$$

- exception: $exc := (s < 0 \Rightarrow OutOfRange | filter\_name \notin \{ramp, None, shepp\} \Rightarrow NULL)$

6.4.5 Local Functions

None

7 MIS of FBP Module

7.1 Module

FBP

7.2 Uses

Filter Module

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
reconstruct	sinogram: $\mathbb{R}^{m \times n}$, $theta : \mathbb{R}^n$, filter_name: string, os: int	image: $\mathbb{R}^{os \times os}$	ValueError, EmptyArrayError

7.4 Semantics

7.4.1 State Variables

$filter \in \mathbb{R}^s$: The fourier filter.

7.4.2 Environment Variables

None

7.4.3 Assumptions

Input sinogram are valid 2D frequency spectra.

7.4.4 Access Routine Semantics

reconstruct(sinogram, theta, filter_name, os):

- transition: $filter := sinogram \Rightarrow Filter.get\_filter(sinogram.size, filter\_name)$ (get_filter from Filter Module10)
- output: out :=
  - $\mathcal{F}^{-1}(\mathcal{F}(sinogram) \cdot filter)$

– perform interpolation over angle theta with the preset image size os.

- exception: exc :=

Exceptions	**Description**
$os < 0 \mid\mid theta < 0 \Rightarrow ValueError$	Valid output size and interpolation angle should not be negative.
$sinogram.size < 0 \Rightarrow EmptyArrayError$	The input sinogram is invalid if the input array size is smaller than 0.

7.4.5 Local Functions

None

8 MIS of Sinogram Simulation Module

8.1 Module

Sinogram Simulation

8.2 Uses

None

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
radon	image: $\mathbb{R}^{m \times m}$, preserve_angle: Boolean, theta: $\mathbb{R}^n$	sinogram: $\mathbb{R}^{m \times m}$	None

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

None

8.4.3 Assumptions

Input Image are valid 2D frequency spectra.

8.4.4 Access Routine Semantics

radon(image, preserve_angle, theta):

- output: out := sinogram where
  - Use padded_radon to find out the padded_image and center of the input image
  - sinogram := $padded\_image \Rightarrow \int_{-\infty}^{\infty} padded\_imgae(x\cos(theta)+y\sin(theta), x\sin(theta)-y\cos(theta))dy$
- exception: None

8.4.5 Local Functions

padding_radon(image, preserve_angle):

- transition: None
- output: out :=

  padded_image :=

  - The diagonal length of the padded square image is calculated as: $d = \sqrt{2} \cdot max(image.size)$
  - The required padding size for each dimension is: $|d - s|, \forall s \in image.size$

  center := $\frac{padded\_image[0]}{2}$

- exception: If image is not an 2D array, raise ValueError

9 MIS of IO Handler Module

9.1 Module

IO Handler

9.2 Uses

FBP Module
Sinogram Simulation Module
Services Module
Hardware Handling Module

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
select_service	op: String	None	None
process_service	File, theta: $\mathbb{R}^n$, filter_name: String, os: int, preserve_angle: Boolean	None	?
get_reconstruction	None	reconstruction: $\mathbb{R}^{os \times os}$	?

9.4 Semantics

9.4.1 State Variables

service: Service sinogram reconstructed_image

9.4.2 Environment Variables

None

9.4.3 Assumptions

Input Image are valid 2D frequency spectra.

9.4.4 Access Routine Semantics

select_services(op):

- transition: service := $op =' reconstruction' \Rightarrow 0 \;||\; op =' verification' \Rightarrow 1$
- exception: exc := $op \notin ['reconstruction','verification'] \Rightarrow ValueError$

process_service(file, theta, filter_name, os, preserve_angle):

- transition: use load_image to load the image data as a 2D array.

  sinogram := $service = 0 \Rightarrow load\_image(file, preserve\_angle) \;||\; service = 1 \Rightarrow simulation.radon()$

  reconstructed_image := fbp.reconstruct() from module...

- exception: exc := ?

get_reconstruction(None):

- output: out := reconstructed_image
- exception: exc := None

9.4.5 Local Functions

load_image(file, preserve_angle):

- output: read image file and return a 2D numpy array.
- exception: exc := file does not exit then FileNotFound.

10 MIS of Configuration Module

10.1 Module

Configuration

10.2 Uses

None

10.3 Syntax

10.3.1 Exported Constants

VERIFICATION = 0
RECONSTRUCTION = 1

10.3.2 Exported Access Programs

None

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

None

10.4.3 Assumptions

None

10.4.4 Access Routine Semantics

None

10.4.5 Local Functions

None

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

11 Appendix

[Extra information if required —SS]